VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2010 **Bc. Jaroslav Smutník**

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

IS pro obchod v MONO

IS for Shop Using MONO

2010

Bc. Jaroslav Smutník

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jaroslav Smutník**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: IS pro obchod v MONO
IS for Shop Using MONO

Zásady pro vypracování:

Cílem práce je vytvořit Client-Server informační systém pro obchod. Systém bude umožňovat evidenci zboží, tvorbu účtenek, faktur, přijemek atd. Systém bude realizovaný v .NET frameworku s přihlédnutím k multiplatformosti a kompatibility v MONO frameworkem.
Struktura práce:
1. Navrhněte univerzální IS pro obchod.
2. Popište MONO framework v kontextu platformy .Net.
3. Realizujte serverovou aplikaci, pro komunikaci použijte Web-Service.
4. Realizujte klienta jako MVC aplikaci.
5. Realizujte klienta pro mobilní zařízení.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Platoš**

Datum zadání: 20.11.2009
Datum odevzdání: 07.05.2010

doc. Dr.Ing. Eduard Sojka
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Súhlasím so zverejnením tejto diplomovej práce podľa požiadaviek čl. 26, odst. 9 *Študijného a skúšobného rádu pre štúdium v magisterských programoch VŠB-TU Ostrava.*

V Ostrave 7.5.2010 .....

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave 7.5.2010 .....

Rád by som poďakoval pánovi Ing. Jánovi Platošovi, vedúcemu mojej diplomovej práce, za jeho cenné, užitočné rady a podnetné pripomienky, ktoré mi ochotne poskytol pri konzultáciách. Taktiež za podklady, ktoré mi poskytol.

Tiež by som sa chcel poďakovať pánovi Ladislavovi Madajovi, ktorý so mnou konzultoval návrh informačného systému z pohľadu zadávateľa v praxi. A tiež v neposlednom rade svojej rodine a priateľke za podporu pri štúdiu.

Abstrakt

Diplomová práca sa zaoberá vytvorením informačného systému pre obchod so skladom v .NET Framework s prihliadnutím k multiplatformosti a kompatibilite s MONO Framework. Zameriavame sa hlavne na poskytnutie funkcií užívateľom, tak aby spĺňali ich požiadavky a uľahčili im prácu. Teoretická časť sa orientuje na opísanie technológií, ktoré boli využité pri vyvíjaní informačného systému. Súčasťou tejto časti je opis možnosti prenosu aplikácie z .NET Framework na Mono Framework. Ďalej obsahuje špecifikáciu požiadaviek, ich analýzu a celkový návrh vytvoreného systému. V praktickej časti je popísaná realizácia samotného informačného systému, ktorý spĺňa architektúru Klient–Server. Zaoberá sa vytváraním dvoch druhov klientov, jedným z nich je klient pre mobilné zariadenia a druhým je klient s možnosťou použitia na dotykové displeje. Cieľom tejto diplomovej práce je vytvorenie nezávislej aplikácie na operačnom systéme, poskytujúcej komfort pri užívaní v oblasti obchodu.

Kľúčové slová: Analýza, Návrh, .Net Framework, Mono, Mobilné zariadenia, Dotykový displej, Informačný systém, Klient –Server, UnIS, Apache, C#, Prenos, PDFsharp, MigraDoc, MVC

Abstract

The diploma thesis deals with the development of the computer information system for the shop with the stock in .NET Framework with regard to multi-platform and compatibility with Mono Framework. We focus especially on offering functions to users to meet their requirements and make their work easier. The theoretical part is orientated on the description of technologies which were used during the development of the computer information system. This part covers the explanation of the transfer of application from .NET Framework to Mono Framework. It includes also the specification of requirements, their analysis and the design of created system. In the practical part, the realization of the computer information system, based on architecture client – server, is described. It deals with creating of two types of clients, one of theme is client for mobile devices and the second of them is client with the option of using on touch screen. The purpose of the thesis is to develop the application independent on operating system and to provide the user's comfort in the business sphere.

Key words: Analysis, Design, .Net Framework, Mono, Mobile devices, Touch screen, Information system, Client –Server, UnIS, Apache, C#, Transfer, PDFsharp, MigraDoc, MVC

Zoznam použitých skratiek a symbolov

atď.	– a tak ďalej
API	– Application programming interface
BCL	– Základná knižnica tried
CAS	– Code Access Security
CLI	– Common Language Infrastructure
CLR	– Common Language Runtime
DA	– Desktop application
FCL	– Framework Class Library
GNOME	– GNU Network Object Model Environment
IDE	– Integrated Development Environment
ID	– Interface Definition Language
LINQ	– Language Integrated Query
ECMA	– European Computer Manufacturers Association
MA	– Mobile application
MoMA	– Mono Migration Analyzer
MSSQL	– Microsoft SQL Server
MVC	– Model–View–Controller
NGWS	– Next Generation Windows Services
ODBC	– Open Database Connectivity
OMG	– Object Management Group
PDA	– Personal digital assistant
RAD	– Rapid Application Development

SQAP	– Simple Object Access Protokol
SQL	– Structured Query Language
SQL CE	– Microsoft SQL Server Compact
UML	– Unified Modeling Language
VB	– Visual Basic
VB .NET	– Visual Basic .NET
VS .NET	– Visual Studio .NET
UnIS	– Univerzálny informačný systém pre obchod a sklad
XML	– eXtensible Markup Language

Obsah

Zoznam tabuliek

Zoznam obrázkov

Zoznam výpisov zdrojového kódu

1 Úvod

Už od nepamäti je snahou našej spoločnosti naplno využiť všetky ponúkané informačné prostriedky a tým si uľahčiť prácu a prispieť tak k zlepšeniu poskytovaných služieb. Veď už i Babylončania si uvedomovali, že ľudský mozog nedokáže byť natoľko dokonalý a výkonný, aby vedel spracovať také množstvo informácii, ktoré ho neustále zahlcujú. Práve s nevyhnutnosťou efektívneho a rýchleho spracovávania informácii prichádza potreba vytvoriť systém, ktorý bude spĺňať požiadavky súčasnej spoločnosti na univerzálnosť, dostupnosť a výkonnosť.

Rozvoj nových technológií zasahuje aj do nami spracovanej oblasti obchodu, kde informačné systémy zaručujú poskytovanie kvalitných služieb, a tým dosahujú spokojnosť nielen u svojich zákazníkov ale i užívateľov. V rámci vzostupu tejto oblasti sa neustále menia nároky firiem na spôsob spracovávania informácií a na používané technológie. V súčasnosti je jedinou a optimálnou cestou využívanie informačných systémov, ktoré sú schopné evidovať, spracovať a analyzovať toto obrovské množstvo údajov.

V našej práci sa preto zameriavame na popísanie technológií použitých pri vytváraní univerzálneho informačného systému, ktorý umožní svojim používateľom efektívne spracovávať potrebné informácie. Cieľom práce je navrhnúť nezávislý a dostupný informačný systém pre obchody so skladom a tým firme zabezpečiť bezproblémovú evidenciu jednotlivých údajov. Našou snahou je vytvoriť systém univerzálny nielen z hľadiska jeho používania ale i z hľadiska nasadenia na rôzne operačné systémy. Zameriavame sa preto najmä na vytvorenie projektu v .NET Framework, ktorý bude kompatibilný s platformou Mono Framework.

Diplomová práca je rozdelená do piatich hlavných kapitol, ktoré popisujú jednotlivé kroky pri vytváraní systému, od priblíženia použitých technológií až po nasadenie aplikácie na vybrané operačné systémy, ako Microsoft Windows, Linux a Windows Mobile.

V prvej kapitole sa zameriavame na poskytnutie všeobecných informácií o platforme Mono Framework z hľadiska histórie a komponent, ktoré ju tvoria. Súčasťou tejto kapitoly je i opis technológie .NET a platformy .NET Framework. Snahou tejto kapitoly je poukázať na možnosti prenosu aplikácií medzi jednotlivými platformami.

Druhá kapitola sa zaoberá špecifikáciou nami vytvoreného informačného systému UnIS. Obsahuje náhľady z funkčného a dynamického hľadiska, ktoré vychádzajú zo špecifikácie. Do tejto kapitoly je zahrnutá analýza a návrh systému.

Samotnou realizáciou aplikácie sa zaoberá tretia kapitola, v ktorej opisujeme implementáciu klientskej a serverovej časti. Klientska časť pozostáva z dvoch typov klientov, a to mobilného a desktopového. U desktopového klienta sme sa zamerali najmä na jeho použitie na dotykových displejoch.

V štvrtej kapitole približujeme postup pri prenose WinForms aplikácií za pomoci Mono Migration Analyzéru.

Posledná kapitola popisuje možnosti nasadenia nášho informačného systému UnIS na spomenuté operačné systémy Microsoft Windows, Linux a Windows Mobile. Pozornosť sme venovali najmä nasadeniu na operačný systém Linux, pretože sa jedná o zložitý proces s využitím serveru Apache.

Dúfame, že naša práca prispeje k uľahčeniu spracovávania informácií prostredníctvom nami vytvoreného systému UnIS a zároveň k spokojnosti všetkých jeho užívateľov.

2 Čo je projekt MONO

Mono je open source softvérová platforma vedená firmou Novell. Prvoradou úlohou bolo vytvoriť a navrhnúť prostredie a nástroje na uľahčenie práce vývojárom a vývojovým týmom pri vytváraní multiplatformných aplikácií aj v inom operačnom systéme ako je Windows. Teda vyvinúť sadu nástrojov kompatibilných s prostredím Microsoft .NET Framework, ktorý podporuje štandardy ECMA(European Computer Manufacturers Association) [1]:

- **Ecma – 334**: C# Language Specification – medzinárodný štandard špecifikujúci formu a zavádzanie interpretácie programov napísaných v C# programovacom jazyku [5],

- **Ecma – 335**: Common Language Infrastructure (CLI) – medzinárodný štandard definujúci základnú jazykovú infraštruktúru (CLI) pre aplikácie napísané v niekoľkonásobne vyšších jazykoch, ktoré môžu byť spustené v rôznych systémových prostrediach bez potreby akokoľvek tieto aplikácie upravovať [6].

Spoločnosť Novell tak chce prijatím štandardizovanej softvérovej platformy znížiť prekážky pri vývoji veľkých aplikácií výhradne pre Linux.

2.1 Niečo z histórie

Pôvodným majiteľom projektu Mono bola firma Ximian, ktorej zakladateľom bol Miguel de Icaza. Vo vnútri spoločnosti prebiehali diskusie o vytvorení nástrojov pre zvýšenie produktivity, čo malo viezť k vytvoreniu viacerých aplikácií s väčšou efektivitou, nižšou námahou a nižšími nákladmi pri tvorbe. Spoločnosť po vyhodnotení štúdie o možnosti vytvorenia takýchto nástrojov vytvorila tým, ktorý mal začať budovať projekt Mono. Účelom týmu bolo využiť .NET Framework pre svoj účel a poskytnúť tak open source Mono.

Oznámenie o tomto projekte sa uskutočnilo 19. Júja 2001 na konferencii O'Reilly. Projekt bol odštartovaný a jeho prvá stabilná verzia Mono 1.0 vyšla o tri roky neskôr 30. Júna 2004.

Logo charakterizujúce Mono predstavuje opičiu tvár. Tento názov vychádza zo vzťahu k opiciam a ľudoopom, ktorý má spoločnosť Ximian. Podobné logá dala spoločnosť svojím projektom ako GNOME projekt, ktorého logo tvorí opičia stopa. Alebo projekt BONOBO, ktorého názov vychádza z druhu šimpanzov a ďalšie.

V nasledujúcej tabuľke môžete nájsť údaje o vývoji jednotlivých verzií až dodnes.

Dátum	Verzia	Poznámky
2004-06-30	Mono 1.0	Podporuje C# 1.0
2004-09-21	Mono 1.1	
2006-11-09	Mono 1.2	Podporuje C# 2.0
2008-10-06	Mono 2.0	Podporuje C# 3.0
2009-01-13	Mono 2.2	Podporuje SIMD
2009-03-30	Mono 2.4	
2009-12-15	Mono 2.6	
Plánovaná verzia	Mono 2.8	Plánovaná podpora C# 4.0

Tabuľka 1: Verzie Mono Framework

V súčasnosti je aktuálna verzia Mono 2.6.3, ktorá poskytuje API z . NET Framework. Táto verzia obsahuje podporu pre verzie jazykov Visual Basic.NET a C # 2.0, 3.0 a 4.0. Umožňuje distribúciu LINQ na XML objekty, no nie na SQL. Kompilátor C# je teraz predvolene nastavený na podporu verzie C# 3.0. Podpora žiadaných Windows Forms je do verzie 2.0. Chystaná podpora verzie C# 4.0 je funkčná už od decembra minulého roka, ale ešte nebola vydaná stabilná verzia.

Vytváranie softvéru na .NET Framework verziu 3.0 je vo vývoji a spadá do experimentálneho podprojektu Mono, nazývaného Olive.

Moonlight

Pre vytváranie aplikácií podobných Microsoft Silverlight , bol vytvorený Moonlight. Podpora tohto podprojektu Mono v rámci Silverlight, bola spočiatku od verzie Moonlight 1.0, ktorý podporuje API Silverlight 1.0 od 20. januára 2009. Novšia verzia Moonlight 2.0 podporuje Silverlight 2.0 a zopár funkcií zo Silverlight 3.0. Začiatkom tohto roku bola ohlásená verzia Moonlight 3.0, ktorá podporuje Silverlight 3 [10], [17], [13].

2.2 Komponenty tvoriace Mono

Medzi základné komponenty tvoriace projekt Mono patria:

- **Kľúčové komponenty** – zahrňujúce spomínaný jazyk C#, jeho gramatiku, sémantiku a základnú jazykovú infraštruktúru. Tieto komponenty sú založené na štandardoch Ecma-334 a Ecma-335, ktoré umožňujú projekt Mono stanoviť ako open source,

- **Mono / Linux / GNOME vývoj** – poskytuje užívateľom podporu a aplikácie pre tvorbu softwaru za plného využitia European Computer Manufacturers Association (GNOME)

a dostupnej knižnice. Medzi tieto nástroje patria: Gtk# pre vytváranie grafického rozhrania, knižnice Mozilla, integrované knižnice Unix, knižnice na pripojenie k databáze a schéma XML jazyk. Na integrovanie Mono aplikácií do GNOME slúži Gtk#. Databázové knižnice poskytujú pripojenie k MySQL, SQLite, PostgreSQL, Firebird, Open Database Connectivity (ODBC), Microsoft SQL Server (MSSQL), Oracle a k mnoho ďalším,

- **Kompatibilita s Microsoft –** umožňuje umiestňovanie Windows .NET aplikácií na GNU/Linux. Táto skupina zahŕňa komponenty ADO.NET, ASP.NET a Windows.Forms.

Base Class Library – Mono Framework poskytuje užívateľom kompletnú sadu tried, ktoré zabezpečujú spoľahlivú funkcionalitu pri vytváraní aplikácií. Tieto triedy sú kompatibilné s triedami od Microsoft .NET Framework.

Mono Class Library – Mono navyše poskytuje mnoho tried, ktoré sú mimo spomínanej základnej knižnice, teda rozširujú funkcionalitu aj nad rámec tried Microsoft .NET Framework. Táto rozšírená funkcionalita je využiteľná hlavne pri vývoji linuxových aplikácií. Ako napríklad triedy pre Gtk +, Zip, LDAP, OpenGL, Cairo, POSIX, a mnoho ďalších [20].

2.3 Výhody

Medzi základné výhody Mono Framework pri vytváraní aplikácií patria:

- **Popularita --** základom je úspech .NET Framework, na ktorom je projekt postavený. V súčasnosti sú milióny vývojárov, ktorý majú záujem vytvárať aplikácie práve v jazyku C#. V tejto oblasti sú vydávané desiatky tisíc kníh a návodov na podporu pri vývoji a tiež nemalým zdrojom sú webové stránky poskytujúce taktiež rôzne príklady a pomoc od ostatných užívateľov,

- **Vyššia úroveň programovania** – všetky jazyky, ktoré Mono podporuje, ťažia z funkcií ako napríklad automatická správa pamäte, reflexia, správa vlákien a mnoho iných funkcií uľahčujúcich prácu. Vďaka tomu umožnia lepšie sa sústrediť na vlastnú tvorbu a nemusíte strážiť infraštruktúru systémového kódu,

- **Základná knižnica tried** – s využitím základnej knižnice tried poskytuje mnoho funkcií zvyšujúcich produktivitu a znižujúcich námahu. Mnoho funkcionality je zahrnutej priamo v platforme a nemusíte tak vytvárať vlastnú,

- **Otvorená platforma** – projekt Mono je vyvíjaný na podporu viacerých platforiem. Beží na Linuxe , Microsoft Windows, Mac OS X, BSD a Sun Solaris, Nintendo Wii, Sony PlayStation 3, Apple iPhone. To tiež beží na x86, x86-64, IA64, PowerPC, SPARC (32), ARM, Alpha, S390, s390x (32 a 64 bitov) a ďalšie. Vývojom svojej aplikácie prostredníctvom MONO zaistíte kompatibilitu s takmer každým počítačom,

- **Common Language Runtime (CLR) –** umožňuje využiť na implementáciu jazyk, v ktorom sa pracuje najlepšie a spojiť ho s iným CLR jazykom. Napríklad, môže dediť VB.Net z vytvorenej triedy v C#, a použiť to v Eiffel. Teda je možné napísať kód v Mono prostredníctvom rôznych programovacích jazykov [20].

2.4 Technológia .NET

.NET („dotnet“) predstavuje úplne nový rámec pre vytváranie veľa druhov aplikácií. Tie zahrňujú aplikácie Windows a webové aplikácie. Tiež sa dá využiť k vývoju systémov skladajúcich sa z prepojených služieb, ktoré pomocou internetu vzájomne komunikujú.

Môže sa navyše použiť aj pre tvorbu aplikácií pre prenosné zariadenia, ako sú prenosné počítače, alebo celulárne telefóny. Iné jazyky tiež umožnia vývoj týchto aplikácií, no .NET bol vytvorený s dôrazom na prepojené siete.

Skladá sa z troch hlavných častí:

- **Vývojové jazyky a nástroje** – vývojové jazyky, s ktorých pomocou je možné vytvárať aplikácie, zahrňujú Visual Basic .NET (VB .NET) a prepracovaný jazyk C++. Microsoft má tiež nástroj pre rýchly vývoj aplikácií Rapid Application Development (RAD) nazvaný Visual Studio .NET (VS .NET). Ten umožňuje vyvíjať programy v integrovanom vývojovom prostredí Integrated Development Environment (IDE). V tejto diplomovej práci budeme používať jazyk C#,

- **Spoločné jazykové prostredie (CLR)** – CLR riadi bežiaci kód a poskytuje rôzne služby, ako správu pamäti, správu tokov (tá umožňuje prevádzať niekoľko úloh paralelne) a vzdialené riadenie (umožňuje, aby objekt v jednej aplikácií komunikoval s objektmi v inej aplikácií). CLR tiež zaisťuje bezpečnosť a presnosť vášho kódu a zabraňuje vzniku chýb,

- **Rámcová knižnica základných tried** – táto knižnica predstavuje širokú skupinu programov, napísanú Microsoftom. Medzi inými obsahuje napr. programy, ktoré dovolia vyvinúť aplikácie pre Windows, pristupovať k adresárom a súborom na disku, komunikovať s databázou a prijímať a odosielať dáta po sieti [22].

2.5 Platforma .NET Framework

Platforma Microsoft .NET Framework prináša celú radu nových konceptov, technológií a pojmov. Jedná sa o softvérový Framework, ktorý môže byť inštalovaný na počítače s operačnými systémami Microsoft Windows.

Základná knižnica tried poskytuje širokú škálu funkcií, vrátane používateľského rozhrania, prístup k dátam, pripojenie k databázam, šifrovanie, vývoja webových aplikácií a komunikácie v sieti. Túto triedna knižnica sa dá využiť a kombinovať s vlastným kódom.

.NET Framework zahŕňa aj verzie pre mobilné zariadenia. Redukovaná verzia .NET Compact Framework, je k dispozícii na Windows CE platformách vrátane Windows Mobile zariadení.

2.5.1 Základné prvky

Interoperabilita

Pretože interakcia medzi novými a staršími aplikáciami je bežne požadovaná, .NET Framework poskytuje prostriedky pre prístup k funkcionalite, ktorá je implementovaná v programoch vykonávaných mimo prostredia .NET. Prístup ku COM komponentom je uvedený v System.Runtime.InteropServices a System.EnterpriseServices menného priestoru, na prístup k ďalším funkciám používame P/Invoke.

Common Runtime Engine

Jedná sa o Common Language Runtime (CLR), teda zložku virtuálneho stroja .NET Framework. Všetky .NET programy sa spúšťajú pod dohľadom CLR, ktorý zaručujúce určité vlastnosti a správanie v oblasti správy pamäte, bezpečnosti a ošetrovania výnimiek.

Jazyková nezávislosť

.NET Framework zavádza spoločný typový systém, alebo CTS. CTS špecifikácia definuje všetky možné dátové typy a programovacie jazyky podporované CLR a ich vzájomné pôsobenie v súlade so špecifikáciou Common Language Infrastructure (CLI). Vďaka tejto funkcií podporuje .NET Framework zmenu typov a inštancií objektov medzi knižnicami a aplikáciami, napísanými pomocou ľubovoľného jazyka vyhovujúceho .NET Framework.

Základná knižnica tried

Základná knižnica tried (BCL) je funkčne dostupnou knižnicou pre všetky jazyky používajúce .NET Framework a je súčasťou Framework Class Library (FCL). BCL poskytuje množstvo funkcií zahŕňajúcich napríklad čítanie zo súboru a zápis do súboru, zobrazovanie grafického rozhrania, prácu s XML dokumentmi, a množstvo iných [22].

2.5.2 Prenositeľnosť aplikácií v .NET Framework

Znamená, že program napísaný pre použitie v .NET Framework by mal bežať bezo zmeny na ľubovoľnom type systému, pre ktorý je daný rámec implementovaný. Aj keď Microsoft nikdy neimplementoval celý rámec do žiadneho systému, s výnimkou Microsoft Windows, je rámec

navrhnutý tak, aby umožňoval implementácie medzi platformami, dostupnými medzi inými operačnými systémami [22].

2.5.3 Architektúra .NET Framework

Common Language Infrastructure (CLI)

Cieľom CLI je poskytnúť jazykovo nezávislú platformu pre vývoj a beh aplikácií zahŕňajúcu funkcie pre riešenie výnimiek a ochrany. Funkcie .NET Framework v rámci CLR nie sú viazané na jeden jazyk, ale na všetky jazyky, ktoré .NET Framework podporuje.

Zlučovanie riadiacich modulov do assembly

CLR vlastne nepracuje s modulmi, ale s takzvanými assembly. Jedná sa o logické zoskupenie jedného, alebo niekoľkých riadených modulov, či zdrojových súborov. Je to najmenšia možná jednotka, podporujúca koncept opakovanej použiteľnosti, bezpečnosti a vytváranie rôznych verzií.

Vďaka assembly je možné zrušiť logické a fyzické väzby znovu použiteľných komponent s možnosťami vytvárať rôzne verzie. Už záleží len na tom, ako sa bude členiť kód a zdroje do rôznych súborov. Samostatné súbory sú stiahnuteľné podľa vlastnej potreby z internetu. V prípade, že nebudete tieto súbory nikdy potrebovať, nebudú stiahnuté a ušetrí sa tak miesto na disku.

Každé vytvorené assembly, môže predstavovať buď spustiteľnú aplikáciu, alebo DLL obsahujúce sadu typov, určených k použitiu spustiteľnou aplikáciou. CLR pritom dohliada na prevádzanie kódu, obsiahnutom v tomto assembly. To znamená, že hostiteľský stroj musí mať nainštalovaný .NET Framework.

Bezpečnosť

.NET ponúka vlastné ochranné mechanizmy a to:

- Code Access Security (CAS),
- Validácia, verifikácia.

Predmetom verifikácie je preskúmanie vysokoúrovňového kódu IL, s cieľom zaistiť, aby všetky prevedené operácie boli bezpečné. Pri verifikácií sa napríklad kontroluje, či sa nečíta nič z pamäte, do ktorej predtým nebolo nič uložené, že každá metóda sa volá so správnym počtom parametrov, každý parameter je správneho typu, správne sa používajú hodnoty vrátené metódami a podobne.

Knižnica tried .NET Framework

Súčasné prostredie .NET Framework je celá zostava assembly .NET Framework Class Library (FCL) obsahujúcich tisícky definícií typov, pričom každý z týchto typov ponúka určité funkcie. Ak to zhrnieme, potom kombináciou FCL s CLR vzniknú aplikácie rôzneho druhu.

Windows.Forms je využívaný pri našej tvorbe:

- **Formuláre Windows (Windows.Forms)** – sú aplikácie so zložitejším grafickým rozhraním Windows. Pri návrhu rozhrania je poskytovaná bohatá funkcionalita aplikácií Windows. Aplikácie Windows Forms s obľubou využívajú ovládacie prvky, ponuky, udalosti myši a klávesnice a majú možnosť priamej komunikácie s operačným systémom. Je tu možné predávať rôzne otázky nad databázou, alebo volať webové služby XML.

Pre využívanie konkrétnej funkcie platformy .NET Framework, je nutné poznať menný priestor, v ktorom sa nachádza. Máte možnosť odvodzovať správanie niektorých typov pomocou dedenia.

Preklad zdrojového kódu do riadených modulov

Keďže sme sa rozhodli vytvoriť aplikáciu postavenú na platforme .NET Framework, našou prvoradou úlohou teda bolo rozhodnúť, aký typ aplikácií budeme vytvárať. Pre našu aplikáciu sme vybrali jazyk, v ktorom bude projekt realizovaný a to C#.

Prekladače

Väčšina prekladačov v minulosti generovala kód pre špecifickú platformu, ako napríklad x86, Alpha alebo PowerPC. Avšak dnešné prekladače, ktoré spĺňajú štandard CLR, generujú kód medzijazyka IL. Tento kód IL môžeme označiť za riadený kód, čo vychádza z CLR, ktoré riadi jeho dobu života aj samotné prevedenie.

Okrem samotného IL musí prekladač pre CLR generovať pre každý riadený modul aj plné metadáta. Metadáta sú sady dátových tabuliek, ktoré popisujú na čo riadený modul odkazuje. Tiež sa dajú metadáta chápať ako nadmnožina starších technológií, ako typových knižníc a súborov v jazyku pre definíciu rozhraní (IDL – Interface Definition Language). Metadáta sú v skutočnosti vždy zapuzdrené v rovnakom súbore EXE/DLL, ako programový kód. To znamená, že vlastný kód sa nedá rozdeliť od metadát. Pre metadáta existuje celá rada využití ako:

- Visual Studio .NET využíva metadáta pre uľahčenie práce programátorovi. Funkcia nazvaná IntelliSense prechádza metadáta a tak umožňuje vytváranie nápovede pri vytváraní metód a typov, prípadne napovedá aké parametre daná metóda očakáva,

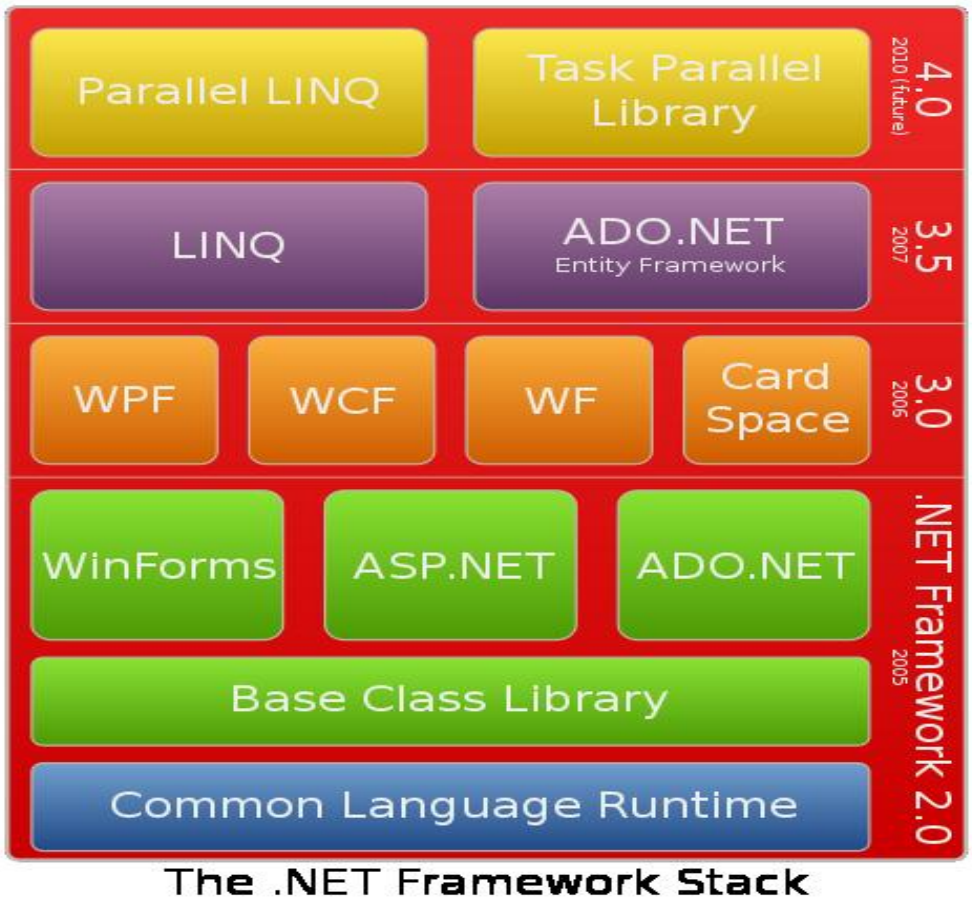
- Automatický správca pamäti využíva metadáta k tomu, aby sledoval dobu života vytvorených objektov,

- Proces overovania správnosti kódu využíva metadáta k tomu, aby sa zabránilo prevádzaniu nebezpečných operácií „verifikácia“.

Prekladač C# spoločnosti Microsoft vždy generuje riadené moduly, ku ktorých spusteniu je potrebný CLR. Koncový užívateľ musí mať na svojom počítači nainštalované CLR [24].

2.5.4 Verzie

Microsoft začal vývoj .NET Framework v druhej polovici roku 1990, pôvodne pod názvom Next Generation Windows Services (NGWS). Koncom roka 2000 bola uvoľnená prvá beta verzia .NET 1.0 [24].



Obrázok 1: Architektúra .NET Framework

[http://upload.wikimedia.org/wikipedia/commons/d/d3/DotNet.svg/]

2.6 Prenositeľnosť aplikácií

Táto časť sa venuje praktikám, ktoré sú užitočné pri vytváraní softvéru, prenositeľného medzi Windows a Unix systémami pomocou. NET a Mono.

Pre začínajúcich vývojárov v Mono a Unix je poskytovaný kompletný image SUSE Linux Enterprise Desktop 10 vrátane najnovšej verzie Mono, vývojových nástrojov, dokumentácie a príkladov na Windows.Forms a ASP.NET. Dostupný je na oficiálnej webovej stránke projektu Mono: http://www.go-mono.com/mono-downloads/download.html.

Súčasťou image je MonoDevelop IDE pre vývoj. Postgress a MySQL sú nainštalované tiež, aby ste si mohli odskúšať svoje aplikácie s databázou [8].

2.6.1 Stratégie prenosu

Dnes Mono obsahuje základné vývojárske knižnice a vývojové nástroje. Väčšina z nich sú nástroje pre príkazový riadok a kompilátor, nejedná sa teda o úplnú náhradu za Visual Studio pre vytváranie Windows.Forms a ASP.NET aplikácií čisto na Linuxe.

Existuje množstvo rôznych spôsobov, ktoré sa dajú využívať pri vytváraní vlastných aplikácií.

Vyvíjanie na Windows a spustenie na Unix

Pre vývoj je možné aj naďalej používať Microsoft Visual Studio pre vytváranie aplikácií na operačnom systéme Windows. Podstatné je, že vytvorené binárne súbory sú kompatibilné s Mono. Potom stačí vložiť tieto súbory do vášho Linux/Unix serveru.

Spôsobov, ako tieto binárne súbory preniesť je viacero. Jedným z nich je ručne skopírovať súbory (adresáre skopírovať pomocou jednoduchého príkazu XCopy), alebo v ideálnom prípade mať vytvorené zdieľanie po sieti medzi Unix a Windows:

- Tak, aby sa dalo pripojiť a zdieľať súbory z Linuxu na Windows,
- Tak, aby sa dalo pripojiť a zdieľať súbory z Windows na Linux.

Dôležité je, že Mirosoft Visual Studio môže vyprodukovať „čiastočne“ spustiteľné súbory, to sú spustiteľné súbory, ktoré nemajú niektoré zo svojich tabuliek komprimované, a sú využiteľné len vo Visual Studio pri vývoji.

Tieto zostavy nebudú správne pracovať s Mono, pretože nie sú podporované. Preto sa musí vypnúť vo Visual Studiu vytváranie „čiastočne“ spustiteľného kódu.

Táto voľba sa dá nastaviť v konfigurácii projektu Visual Studia.

Vývoj čisto na systéme Linux

Existuje v tejto oblasti niekoľko IDE, ktoré môžu byť použité priamo na Linuxe, s rôznym stupňom funkčnosti:

- **X-Develop** – je komerčný produkt od firmy Omnicore, ktorý poskytuje kompletné multijazyčné a multiplatformové IDE pre profesionálnych vývojárov. Dáva slobodu voľby a podporuje .NET, Mono a Java platformy pre Windows, Linux a Mac OS X. Prichádza s podporou pre C#, Java, JSP, J#, Visual Basic .NET, JavaScript, XML a HTML,

- **MonoDevelop** – IDE vyprodukované komunitou projektu Mono, ktorý má podporu pre vývoj grafických aplikácií s využitím Gtk#. V súčasnosti ešte stále chýba podpora pre Windows.Forms alebo ASP.NET.

Ak je pohodlnejšie a praktickejšie používať nástroje príkazového riadka, potom je možnosť využívať podporu, ktorú Mono týmto nástrojom venuje, a ktoré pracujú spoľahlivejšie ako produkt MonoDevelop.

Všeobecné pokyny

Od verzie Mono 1.1.18 je poskytovaný nový IO Remapping. Jedná sa o funkcie, ktoré zvládajú case sensitive (rozpoznávajú malé a veľké písmená) systému súborov a adresárov v názvoch súborov.

Pomocou IOMap funkcie je možné rýchlo preniesť aplikáciu, ale prenos nie je závislý na tomto premapovaní (avšak má vplyv na zníženie výkonu).

Case sensitive

Linux a Unixové súborové systémy sú case sensitive (rozpoznávajú malé a veľké písmená). Súbory s názvami readme a README sú dva rozdielne súbory.

To je dôležitý rozdiel v mnohých aplikáciách, pretože by ste vytvorili napríklad súbor Login.aspx, ale odkazoval by na login.aspx alebo LOGIN.ASPX zo zdrojového kódu.

Počiatkom verzie Mono 1.1.18, je možné nastaviť prostredie MONO_IOMAP, ktoré v Mono umožní odstrániť problémy s case sensitive.

Oddeľovače

Na Windows sa zapisuje cesta k adresárom a súborom pomocou oddeľovača „\“, zatiaľ čo v systéme Linux je to oddeľovač „/“. Je možné vytvárať súbory, ktoré obsahujú oddeľovače typu Windows aj na Linux.

Opäť je možné využiť prostredie MONO_IOMAP. Teda ak je požiadavka na vytváranie prenosného softvéru, je nutné sa uistiť, že sa používa znak System.IO.Path. DirectorySeparatorChar, keď sa spájajú cesty k adresárom, alebo súborom. Ešte lepším spôsobom je využiť IO.Path.Combine metódu, používanú na kombinovanie ciest.

Ak sa vkladá kód manuálne, bez využitia IOMAP, cesty sa upravujú nasledovne:

```
int index = exePath. LastIndexOf ("\ \");

exeDir = exePath. Substring (0, index);

exeFile = exePath. Substring (index + 1);
```

Cesta a ostané premenné majú adresáre oddeľované pomocou bodkočiarky „;“ na Windows a dvojbodkou „:“ na Linuxe. Mali by sa použiť na oddeľovače System.IO.Path.PathSeparator.

```
Console.WriteLine("Subdirectories found in the PATH environment
variable:");

string path_env = Environment.GetEnvironmentVariable ("PATH");

string[] path_dirs = path_env.Split (Path.PathSeparator);

foreach (string pathdir in path_dirs)

      Console.WriteLine(pathdir);
```

Niekoľko typov ako na mapovanie ciest v Mono:

- Vo všeobecnosti má každý snahu využiť všetky dostupné operácie platformy, na ktorej program vytvárate. Ale pre rôzne platformy môžu byť cesty k súborom realizované ináč. Malé a veľké písmena, PathSeparator, DirectorySeparatorChar, atď. Tu však môže dôjsť k sporom vytváraných programov s nasadením na inú platformu, ako na ktorej bol program vytváraný.

- Pre triedy, ktoré manipulujú s cestami, je užitočné využívať detekciu platformy, pred prácou s cestami a následne využívať potrebné metódy na prácu s cestami pre každú platformu zvlášť.

- Pri niektorých testoch je možné využívať aj cesty typu „/a/b/c“, ktoré budú fungovať na obidvoch platformách. No nesmiete zabudnúť na to, že metódy ako Path.GetAbsolutePath() budú mať odlišné správanie na každej platforme.

Absolútny názov cesty

Používanie absolútnych ciest názvov v aplikáciách vedie k problémom pri nasadení aplikácie z Windows na Linux, kde cesty nebudú fungovať.

Pri potrebe lokalizovať nejaké služby, by malo byť správne použité .NET API. Napríklad na zistenie umiestnenia konfiguračného súboru môže byť použité:

```
AppDomain.CurrentDomain.SetupInformation.ConfigurationFile
```

Vyvolávanie procesov na rôznych platformách

Vo väčšine kódu vyvolávajúceho čisto Windows knižnice bude potrebné upraviť príslušné volania v Linuxe, alebo bude musieť byť volanie úplne upravené a prispôsobené Linuxu.

Použitie registrov

Triedy registrov sú poskytované Linuxom, ale sú užitočné iba pokiaľ odkazujú na konfiguračné nastavenia, ktoré používa aplikácia. Tieto triedy neposkytujú žiadne užitočné informácie o nastavení alebo konfigurácii operačného systému. Mono prevádza kľúčové názvy a ich význam na malé písmená. Normálny prístup nie je závislý na veľkosti znakov, ale ak aplikácia načíta názvy z registra a porovná ich s pevnou hodnotou, musí sa použiť porovnanie znakovo nezávislé. Toto sa často používa pri testovaní kódu, ale i v iných situáciách.

Migrácia „presun“ databázy

Databázu nie je potrebné presúvať, je možné pokračovať v používaní SQL serveru s Mono. V prípade ak sa bude nahrádzať SQL server s inou databázou, umožňuje Mono rozsiahle možnosti databázových spojení pre MySQL, PostgreSQL, Sybase, Oracle, IBM DB2, Firebird, ODBC, a tiež pre vloženú SQLite databázu. Existujúci kód ako aj prepojovacie reťazce budú správne pokračovať v práci, jedinou potrebnou zmenou môže byť zmena hostname databázy.

Chýbajúce funkcie

Mono postráda množstvo funkcií využiteľných v .NET, tu sú niektoré z nich:

- **Žiadne Enterprise.Services** – ak aplikácia vyžaduje tieto služby, softvér pravdepodobne nebude bežať v Mono,
- **Žiadne medziprocesové operácie** – v súčasnosti Mono podporuje len lokálne spracovanie operácií,

- **Žiadne COM** – v Unixe nie je COM súčasťou operačného systému a preto nie je Mono(m) nepodporovaný. Ak aplikácia COM vyžaduje, musia sa zabaliť metódy, ktoré sa volajú pomocou P/Invoke.

Nástroje umožňujúce prenos:

- **MonoDevelop** – verzia MonoDevelop 0.14 umožňuje načítať projekty z Microsoft Visual Studia 2005 a môže projekt skompilovať prípadne generovať Unix súbory spustiteľné prostredníctvom svojho IDE. Po importovaní projektového súboru z Visual Studia 2005 s koncovkou .sln do MonoDevelop a skompilovaní, stačí kliknúť pravým tlačidlom na Solution a vybrať "Generate Autotools Framework",

- **Installvsts** – na inštaláciu rôznych štartovacích balíčkov, dostupných na webovej adrese: http://asp.net, môže sa použiť program installvsts,

- **Prj2make** – je vhodným nástrojom, na presunutie aplikácie na Linux. Mono je dodávané s nástrojom nazvaným Prj2make, ktorý prevádza projekt Visual Studia 2003 na skupinu spustiteľných súborov. Program je schopný previesť okolo 50 % existujúcich riešení na spustiteľné súbory, a je užitočný pri prvom prechode konverzie z projektu na skupinu Unix – spustiteľných súborov, ale nie je to stopercentné. Problém, ktorý Prj2make má, je že nie je schopný kopírovať súbory s rozdielmi v názvoch, takže je možné, že váš projekt odkazuje na súbor Core.cs, zatiaľ, čo súbor na disku je pomenovaný ako core.cs, ak bude kompilácia neúspešná, mal by sa opraviť názvy súborov [8].

2.7 Licenčné riešenie

Licenčná schéma je naplánovaná tak, aby umožňovala súkromným vývojárom písať aplikácie v Mono.

Mono využíva 4 open source licencie:

- C# kompilátor je licencovaný pod dvoma licenciami MIT/X11 a GNU General Public License (GPL),

- Nástroje sú povolené podmienkami stanovenými v GNU General Public License (GPL),

- Runtime knižnice sú pod GNU Library GPL 2.0 (LGPL 2.0),

- Knižnice tried sú povolené licenciou MIT/X11,

- ASP.NET MVC a ASP.NET AJAX klient software sú vydávané Microsoftom pod open source Microsoft Permissive License.

Ak nie je vhodná kombinácia LGPL/GPL/X11, je Mono licencované komerčne. Nástroje Mono pre Visual Studio Ultimate Edition zahŕňajú komerčné licencie na redistribúciu Mono pod inými ako LGPL podmienkami Windows, Linux a Mac OS X PC pre produkty s kapacitou pod 100000 a príjmom pod $2M ročne. The Ultimate Edition je to správne, pokiaľ má organizácia v úmysle distribuovať software, ktorý vloží alebo zbalí do Mono, ale nie je schopná splniť podmienky GNU LGPL v2.

Prečo sú knižnice tried licencované pod MIT/X11?

Pôvodne boli knižnice tried licencované pod podmienkami GNU Library GLP. Problémom GNU Library GLP je zastaraná formulácia týkajúca sa „odvodených prác“. Práca odvodená od knižnice musí byť pokrytá rovnakou licenciou ako knižnica sama. Táto definícia bola v poriadku, pokiaľ neexistoval objektovo – orientovaný rámec, ale s jeho príchodom sa mnoho ľudí nezhodlo, či nejaký kód, ktorý používa objektovo – orientované dedenie je inštanciou „odvodenej práce“. Knižnice tried sú obrovskou súčasťou Mono, ku ktorej prispela najrôznejšia skupina jednotlivcov. Vzhľadom na nejednoznačnosť a možnosť, že každý autor knižníc tried mohol rôzne pochopiť licenčné podmienky, spoločnosť starajúca sa o projekt Mono vybrala licenciu, ktorá stále bola open source/ free software licenciou, ale nemala tieto nejasnosti. Táto nejednoznačnosť by dovolila autorovi kódu nárokovať si od vývojára aplikácie uvoľnenie časti jeho zdrojového kódu, a to Mono nechcelo riskovať.

Kedy môžete získať licenciu od Novelu k využívaniu Mono?

Mono požaduje licencie pre používanie Mono a Moonlight v systémoch, ktoré nie sú schopné splniť povinnosti GNU LGPL. Napríklad ak sa spravuje zariadenie, ktorého koncový užívateľ nie je schopný aktualizovať Mono virtual machine, alebo Moonlight runtime zo zdrojového kódu, je potrebná komerčná licencia Mono a Moonlight.

Prečo Novel požaduje copyright špecifikáciu?

Keď vývojár poskytuje kód C# kompilátoru, alebo Mono runtime, požaduje po autorovi, aby udelil Novelu právu znovu licencovať jeho príspevok pod inými licenčnými podmienkami. To umožňuje Novelu predistribuovať Mono source kód stranám, ktoré možno nebudú chcieť používať GPL alebo LGPL verzie kódu [11].

3 Návrh UnIS pre obchod

Účelom tejto diplomovej práce je navrhnutie univerzálneho informačného systému „UnIS“ a následné zrealizovanie pomocou .NET Framework s prihliadnutím k multiplatformosti Mono Framework. Ako isto viete, .NET sa stáva horúcou platformou pre vývoj a implementáciu budúcnosti a to je jedným z dôvodov, prečo sme sa rozhodli na realizáciu systému použiť práve tento Framework.

Komunikáciu klientskej časti so serverovou budú zaisťovať webové služby. Funkcie serveru sa postarajú o správu dát v pripojenej databáze, ktorá bude bežať na lokálnom, alebo vzdialenom databázovom serveri. Rozšírením systému bude rozdvojenie klientskej časti:

- Klient DA (desktop application),
- Klient MA (mobile application) s podporou Windows Mobile 6.0.

Pri vytváraní klientskej časti ako Model-View-Controller (MVC) aplikácie, bude prihliadané na používanie klienta DA pre dotykové displeje.

3.1 Špecifikácia UnIS

Hlavým cieľom systému bude zautomatizovať agendové spracovanie skladov spolu s predajom a prímom tovaru na sklady. Čo sa týka klienta DA, bude realizovaný ako tenký klient, nebude u seba uchovávať dáta v databáze, teda bude pracovať s dátami, ktoré získa len od servera. Naopak klient MA bude obsahovať vlastnú databázu, ktorej úlohou bude zaistiť samostatných chod klienta, bez potreby využívania služieb servera. Tie však bude klient MA pre mobilné zariadenia využívať hlavne k synchronizácií dát, aby sa aplikácia nerozchádzala a mala aktuálny stav databázy.

Užívatelia systému v klientskej časť budú mať pridelené role, ktorým budú odpovedať jednotlivé obmedzujúce nastavenia aplikácie. Pre zlepšenie predstavy o funkcionalite výsledného produktu, boli zostavené scenáre z rôznych pohľadov, charakterizujúcich jednotlivé role systému, ako aj stavy v ktorých sa môže systém nachádzať.

Pohľad Administrátora

Administrátor bude reprezentovať rolu s plným užívateľským prístupom k funkcionalite vytváraného systému UnIS, ako aj správu a udržiavanie agendy firmy. Táto pozícia patrí predovšetkým vedeniu firmy, kde patrí vedúci prevádzky, alebo vedúci celej firmy v závislosti od vnútornej hierarchie firmy.

Medzi základnú úlohu, od ktorej bude záležať počiatočné nastavenie firmy, patrí inštalácia aplikácie a nastavenie jednotlivých ciest pre nadviazanie spojenia klientskej časti so serverom. To znamená nastavenie správneho chodu webových služieb, ako aj pripájacieho reťazca k databáze, ktorú bude spravovať server. Po týchto úkonoch bude administrátor vyzvaný na vytvorenie firmy,

kde sa bude systém používať a ku ktorej sa budú viazať skladové karty, sklady, príjemky, výdajky a ďalšie komponenty v ktorých bude firma vystupovať. Časť server sa automaticky v pozadí postará o zaznamenanie údajov do pripojenej centrálnej databázy s hierarchiou UnIS. Následne program vyzve administrátora, aby vytvoril kľúčového užívateľa systému, ktorý bude mať popisované práva v tomto pohľade, to znamená, plný prístup k funkcionalite aplikácie. Po nainštalovaní UnIS sa užívateľ bude môcť prihlási pod údajmi, ktoré zadal pri inštalácií a bude mu zobrazený mód práce administrátor. V tomto prostredí bude mať k dispozícií vopred vytvorené funkcie na správu agendy firmy. Na karte zamestnancov bude mať možnosť zaevidovať jednotlivých užívateľov do systému a podľa očakávaného zaradenia im prideliť jednu z rolí:

- **Administrátor** – táto rola je popísaná v pohľade administrátora,
- **Predavač** – táto rola je popísaná v pohľade predajcu,
- **Skladník** – táto rola je popísaná v pohľade skladníka.

Režim administrátora bude obsahovať aj funkcie ostatných režimov, popísaných v nasledujúcich pohľadoch ako aj niektoré prioritné funkcie navyše. K prioritnej funkcionalite patrí už spomínaná inštalácia systému a správa užívateľov systému spolu s údajmi o vytvorenej firme.

Umožní užívateľom využívať aj klienta MV pre mobilné zariadenia. Funkcie, ktoré bude poskytovať, sú popísané v pohľade na užívateľa pokladnice pre mobilné zariadenia.

Pohľad skladníka

Registrovaný skladník bude mať také zaradenie užívateľa, ktoré obsahuje o niekoľko funkcií menej ako administrátor. Rozdiel v týchto funkciách bude vychádzať z obmedzenia na správu agendy firmy, teda obmedzenie správy užívateľov a firmy, ako aj iné nastavovacie funkcie systému, dostupné len pre administrátora.

Prioritnou úlohou skladníka bude vytváranie skladových kariet, kde bude možnosť zaznamenania informácií o budúcom predaji. Tieto informácie skladník zadá na základe prepočtov rozpočítavania cien pre predaj. To znamená napríklad zadanie nákupnej ceny pre konštantnú mernú jednotku l (liter) pri rozlievaných nápojoch a následné zvolenie prepočtu základnej mernej jednotky pre predaj, ako podiel z konštantnej mernej jednotky. Teda nákup rozlievaných nápojov bude v litroch a predaj v decilitroch, čo sa odrazí na prepočítanej predajnej cene v pokladni.

Okrem iného bude pokladaná za základnú operáciu aj vytváranie príjemiek a teda týmto spôsobom dodávanie tovaru do skladov. Sklady bude možnosť vytvárať podľa zaradenia a adresy umiestnenia a pridávať im jednotlivé skladové skupiny, ktoré budú obsahovať jednotlivé skladové karty. Na vytváraných príjemkách musí byť vždy evidovaný zaregistrovaný dodávateľ. V prípade, že dodávateľa nenachádza v evidencii firmy, potom bude mať možnosť skladník dodávateľa pridať,

či upraviť jeho stávajúce informácie pri zmene. Pri vymazávaní dodávateľov, ako aj odberateľov a ostatných položiek zo systému, prebieha kontrola závislosti objektov na ostatné tabuľky.

Aby bola možnosť stornovať objednávky, či vyraďovať doklady, systém bude obsahovať funkciu vyraďovacích dokladov s dôvodom vyradenia. Pri vytvorení vyraďovacieho dokladu bude tovar vyradený zo systému a vytvorený záznam o vyradení.

Pre umožnenie tvorby tlačových výstupov zo systému budú mať užívatelia možnosť vytvárať výstupy priamo v programe a tiež tlačiť výstupy priamo z programu bez použitia iných nástrojov na prezeranie dokumentov. Druhou možnosťou bude vytváranie tlačových výstupov do externých súborov s formátom PDF a priamo otvárané programom na prezeranie, či prípadnú tlač.

Po dohode s konzultantom bude ponechaná tomuto režimu aj funkcia vytvárania výdajok zo systému na registrovaných odberateľov v systéme. Takisto bude možnosť registrovať odberateľov na vytvárané doklady, pretože bez nich by nebolo možné výdajku vytvoriť . Pri úprave jednotlivých odberateľov prebieha už spomínaná kontrola správnosti vyplnenia formulára.

Tento režim bude zahrnutý do klienta MV pre prenosné zariadenia. Bude však obmedzený len na predaj z mobilného zariadenia. Klienti v prenosných zariadeniach budú využívaní v rámci reštaurácií a uľahčujúci prácu „čašníkom“, teda vytváranie účtov zákazníkom. O tomto spôsobe predaja sa zmienime viac v pohľade na užívateľa pokladnice pre mobilné zariadenia.

Pohľad predajcu „čašníka“

Režim predajcu bude nastavený predovšetkým na predaj tovaru zo skladu a optimalizovaný na používanie na dotykových displejoch. Tiež bude rozšírený o predaj za pomoci spomínaného prenosového zariadenia.

Aby bol systém univerzálnejší, bude obsahovať dva typy pokladne:

- **Reštauračná pokladňa** – bude vytvorená hlavne pre reštauračný predaj a umožňujúca vytváranie viacerých účtov na mená, alebo iných identifikačných značiek rozlišujúcich jednotlivých zákazníkov. Prichádzajúci zákazník bude mať vytvorený účet na reštauračnej pokladni, do ktorého mu predavač bude prihadzovať jednotlivé položky počas jeho zdržania sa v reštaurácií. Počas celej návštevy bude udržiavaný aktuálny stav účtu a výsledná cena k zaplateniu. Na takýto typ účtu sa nebude zadávať odberateľ a bude možnosť vystavenia daňového dokladu v spomínanom formáte PDF, ktorý po odoslaní na tlačové zariadenie vytvorí výsledný účet. Po odoslaní bude v pozadí upravený počet položiek v skladoch,

- **Pokladňa pre odberateľov „výdajka“** – tento typ bude predstavovať pokladňu v „kamennom“ obchode, alebo vytváranie výdajok zo skladu. Neumožní vytvárať viacero výdajok súčasne a vytváranie výdajok bez odberateľov. Tento druh predaja nebude zahrnutý do prenosného zariadenia z dôvodu spomínanom v prehľade registrovaného skladníka.

Predavač bude mať spolu s režimom administrátora prístup k funkcii elektronickej peňaženky. Táto funkcionalita bude doložená pre uľahčenie a kontrolu vlastnej hotovosti predavača „čašníka“. Princíp bude založený na vkladaní a výbere hotovosti s udaním dôvodu, napríklad ak majiteľ potrebuje hotovosť od zamestnanca a ten si musí upraviť stav v elektronickej peňaženke, aby mu nenastali konflikty pri výslednom súčte. Pri každom predaji sa výsledná suma pričíta automaticky do elektronickej peňaženky a zamestnanec bude vidieť aktuálny stav, ktorý by mala obsahovať jeho fyzická peňaženka, alebo kasa.

Vo všetkých režimoch bude obsiahnutá tvorba uzávierok s troma rôznymi vlastnosťami:

- **Denná uzávierka** – charakterizuje uzávierku predaného tovaru, bude teda pozostávať z vytvorených výdajok, pričom nezáleží akého boli typu (reštauračné, výdajky z prenosného zariadenia, výdajky s odberateľom, výdajky bez odberateľa) a vypočíta uzávierku za celý odpracovaný deň,

- **Mesačná uzávierka** – bude mať rovnakú funkcionalitu ako denná uzávierka, s tým rozdielom, že spracovávané obdobie za predchádzajúci mesiac,

- **Uzávierka s nešpecifikovaným dátumom** – bude druh uzávierky, kde si užívateľ navolí presne obdobie, ktoré chce spracovať a vystaviť.

Systém umožní prehľadávať uzávierky a vytvárať z nich výstupné zostavy priamo v programe, alebo v externom súbore vo formáte PDF.

Užívatelia tejto role môžu využívať aj klienta MV pre mobilné zariadenia, ktorý tak uľahčí ich prácu a to hlavne v oblasti reštauračného predaja. Podrobnejšie sú popísané funkcie nižšie.

Pohľad na užívateľa pokladnice pre mobilné zariadenie

Zaregistrovaný užívateľ, ktorý bude mať možnosť prihlásenia do systému mobilnej pokladnice, bude ktorýkoľvek registrovaný zamestnanec s pridelenými prihlasovacími údajmi v hlavnej databáze systému. Teda mobilná pokladnica bude obsahovať len jednu rolu, pretože operácie, ktoré sú pridelené pokladni, sú v klientovi DA dostupné zo všetkých režimov.

Pred prihlásením do systému bude mať užívateľ možnosť nastavenia reťazca, pre pripojenie k webovým službám, ktoré umožnia komunikáciu so serverom obsahujúcim funkcie pre synchronizáciu mobilného zariadenia s centrálnou databázou. Dôvodom je hlavne prvé spustenie, alebo zmena poskytovateľa webových služieb. Po nastavení reťazca na webovú službu bude v móde neprihláseného užívateľa možnosť otestovať spojenie a zosynchronizovať databázu v mobilnom zariadení s centrálnou databázou, čo sa registrovaných užívateľov týka, teda obnova užívateľov systému s aktuálnymi prihlasovacími údajmi.

Po prihlásení bude mať užívateľ rôzne možnosti ako vytvoriť účet. Jeden spôsob je na meno zákazníka, alebo pomocou jednoznačnej identifikácie, ktorú si vyberie, aby rozpoznal vytvárané účty. Databáza v mobilnom zariadení bude uchovávať aktuálne informácie o vytváraných účtoch a stavy, v ktorých sa účty nachádzajú. Tieto stavy môžu byť tri a podľa nich bude mať užívateľ možnosti a obmedzenia nastavené aplikáciou pre jednotlivé položky. Stav:

- **Nezaplatený** – znamená práve vytváraný účet, alebo inak povedané otvorený účet. V tomto prípade bude aplikácia automaticky sprístupňovať funkcie ako sú vymazanie účtu, zmena údajov o účte, vymazanie položiek účtu, zmena stavu položiek účtu, alebo možnosť prevedenia účtu do stavu zaplatený a tým pádom automatické uzatvorenie vytváraného účtu. V takomto stave môže byť súčasne viacero otvorených účtov, ku ktorým sa môže užívateľ v systéme vracať a meniť údaje,

- **Zaplatený** – bude stav účtu vždy po vykonaní platby. Teda účet bol najskôr vytvorený v stave nezaplatený a následne prevedený platbou automaticky do tohto stavu. Charakteristické pre tento stav účtu bude jeho nedotknuteľnosť v databáze mobilného zariadenia, čo znamená odobratie funkcií zmazania účtu, jeho zmeny a zmeny jeho položiek, ako aj zmazávanie položiek, či prevedenie do iného stavu. Možnosti, ktoré systém ponúkne budú obmedzené len na prezeranie položiek takéhoto účtu,

- **Synchronizovaný** – bude účet až vtedy, keď prihlásený užívateľ prevedie synchronizáciu, ktorá bude bližšie popísaná nižšie. Pri tomto stave sa v účte bude môcť užívateľ len pohybovať, no nebude môcť pracovať s jeho položkami. Funkcia, ktorá mu bude poskytnutá s týmto účtom bude len vymazanie celého účtu aj s položkami, pretože už bol synchronizáciou odoslaný do hlavného skladu.

Pri tvorbe a prehľade účtov bude automaticky tvorená a prepočítavaná výsledná cena k zaplateniu, podľa množstva a ceny poskytovaných produktov. Poskytované produkty si bude môcť užívateľ vybrať zo zobrazeného skladu s možnosťou rýchleho vyhľadávania podľa názvu produktu.

Synchronizácia zariadenia s hlavnou databázou bude dôležitá, pretože vytvorené účty v mobilnom zariadení budú mať vlastnú databázu a nezosynchronizované účty sa nezaevidujú do hlavnej databázy systému UnIS. Druhy synchronizáciu sú tri:

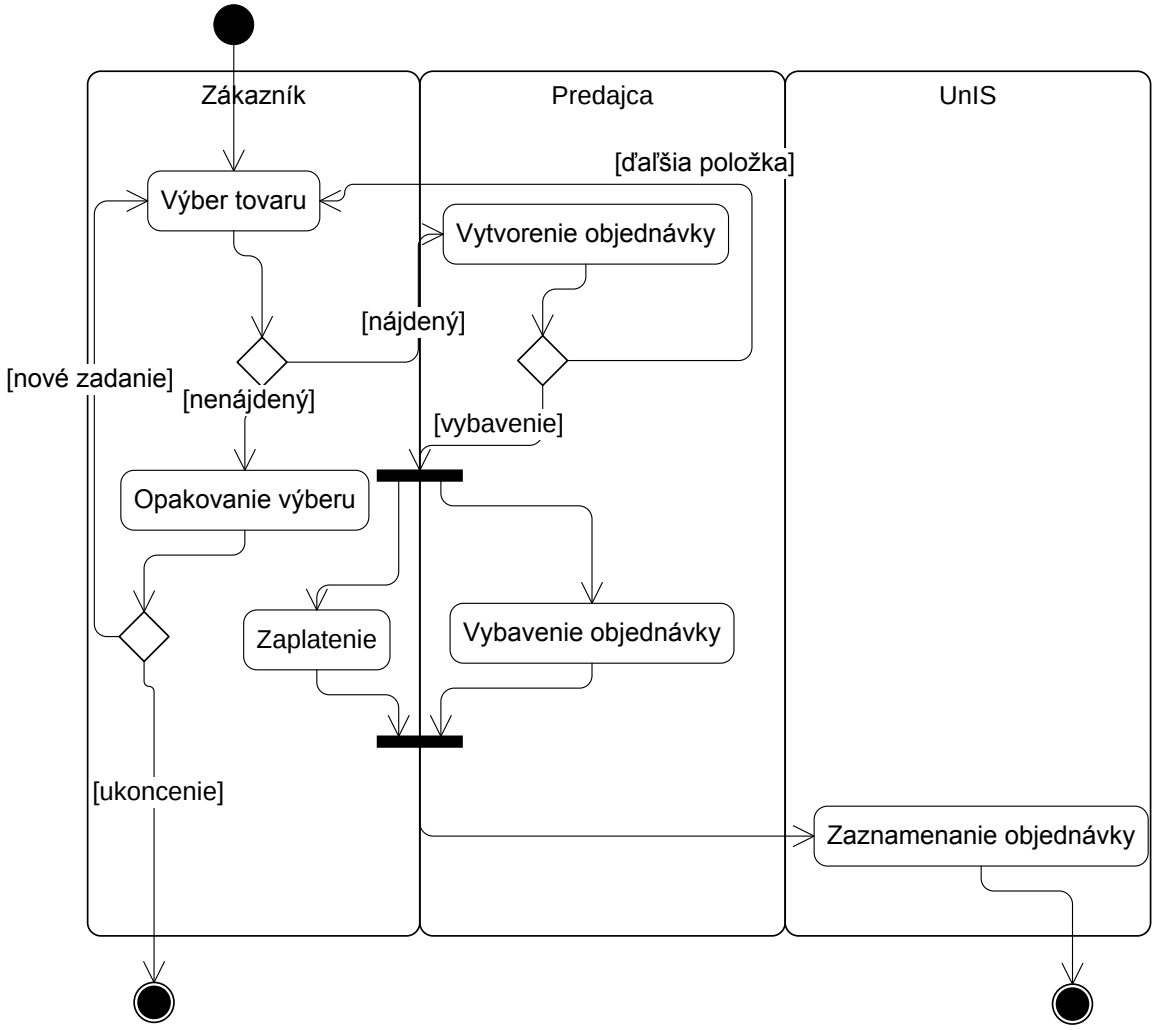
- **Synchronizácia zamestnancov** – spomínaná už v začiatku tejto kapitoly,

- **Synchronizácia skladu** – založená na obnove a aktualizovaní dát v sklade hlavnej databázy a v sklade databázy mobilného zariadenia,

- **Synchronizácia účtov (výdajok)** – po nadviazaní spojenia so serverom zapíše všetky účty z databázy mobilného zariadenia do hlavnej databázy a u týchto zapisovaných účtov v mobilnom zariadení zmení stav zo zaplatený na synchronizovaný.

Preto bude dôležité mať pred každým ukončením práce s mobilným zariadením, alebo v inom zamestnávateľom určenom termíne, vykonanú synchronizáciu mobilných zariadení s hlavnou databázou, čím si budeme udržiavať aktuálny prehľad o tovare na skladoch.

Nasledujúci diagram aktivít reprezentuje základný kameň podnikového procesu a to predaj tovaru zákazníkovi so zaznamenaným priebehom v systéme UnIS.



Obrázok 2: Diagram aktivít predaj tovaru

3.1.1 Rozsah

Špecifikácia systému má za úlohu vytvoriť podklady pre návrh a vytvorenie aplikácie. U vytvoreného systému bude požadovaná komunikácia s databázou riešená pomocou webových služieb. Teda systém pobeží na lokálnej sieti firmy so zabudovaným IIS serverom, alebo budú poskytované serverové služby pre systém prostredníctvom internetu.

Klient DA umožní prihlásenie užívateľov do viacerých rolí a podľa prihlásenia zobrazí užívateľské prostredie na prácu v administrátorskom režime, skladníckom režime, alebo režime predajcu. Jednotlivé funkcie dostupných režimov sú opísané v pohľadoch na užívateľské role.

Klient MA bude komunikovať so serverom tak isto ako klient DA, teda prostredníctvom webových služieb. Systém pobeží na platforme Windows Mobile 5.0 Pocket PC SDK a webové služby bude používať prostredníctvom wireless pripojenia na internet.

Jednotlivé funkcie mobilného klienta sú eventuálne z pohľadu na užívateľa pre mobilné zariadenia.

3.1.2 Prečo UnIS

Zadávateľská firma potrebuje informačný systém z dôvodu zvýšenia prehľadu o tovare v sklade a o predaji na jednotlivých pobočkách. Je potrebné vytvárať doklady o zaplatení, ako aj faktúry pre odberateľov a tieto výstupy zobrazovať v dostupnom formáte pre vytlačenie. Prínosom tejto aplikácie bude uľahčenie práce skladníkov, ktorí budú mať okamžitý prehľad o položkách v skladoch, a tým im systém uľahčí objednávky.

Zatiaľ ešte žiaden informačný systém vo firme nasadený nie je a ručné spracovanie množstva informácií je náročné a príliš pomalé, vyžaduje sa rýchla reakcia systému na prácu s dátami. Užívateľovi by mal systém slúžiť k pohodlnej práci s dátami v databáze, prístup k informáciám by mal byť rozdelený podľa role užívateľa cez klientsku časť.

Podstatou celého UnIS(u) je uľahčenie práce užívateľom, ich zefektívnenie a urýchlenie vyhľadávania tovaru v skladoch ako aj vytváranie jednoznačných príjmových a výdavkových dokladov.

3.1.3 Prečo UnIS pre mobilné zaradenia

Firma zadávajúca projekt má viacero čašníkov, ktorý si vytvárajú zápisky na lístky a nemá automatizované vytváranie účteniek jednotlivým zákazníkom. Týmto systémom chce uľahčiť prácu svojím zamestnancom a znížiť tak nároky na ich pamäť. Tým sa vyvaruje zabudnutiu vytvorenia objednávky, alebo nesprávnemu zarátaniu položiek na účet pri veľkom počte zákazníkov.

Systém vo firme nie je, bude obsahovať systém UnIS s klientom DA, ktorý ale nebude možnosť nosiť zamestnanec so sebou po reštaurácií, pretože sa bude jednať o klienta vo forme desktop application. Dvojité zapisovanie, teda zaznačovanie objednávok pri stole a následne zaznamenanie objednávok na účet v hlavnom klientovi aplikácie, vedie k zbytočnému zdržiavaniu práce a zavádzaniu chýb.

3.1.4 Pohľad na problém zadávateľskej firmy

Nasledujúci problém – firma žiaden univerzálny informačný systém nemá a musí viesť ručnú evidenciu všetkého tovaru a skladov. Pri vytváraní jednotlivých účtov neobsahuje reštauračnú

pokladnicu a užívatelia „čašníci“ pre reštauračný predaj sú odkázaný na svoje poznámkové bloky, ktoré nechávajú na stoloch, alebo u seba. Užívatelia nemajú k dispozícií ani žiaden systém pre mobilné zariadenia, ktoré by im uľahčili prácu namiesto poznámkových blokov, alebo lístočkov.

Pozície, ktorých sa tento problém priamo dotýka - vedenie celej firmy (vedúci), zamestnanci na pozíciách skladníkov a predajcov:

- Pre reštauračný predaj – predavač „čašníci“,
- Pre jednotlivý predaj – predavači v „kamenných“ obchodoch.

Dôsledky – celý proces vytvárania a sčítavania vytváraných účtov pre zákazníkov je časovo náročný a znižuje tak efektivitu zamestnancov, ako aj komfort zákazníkov. Tiež pamäťové nároky na zamestnanca „čašníka“ zo strany klientov sú vysoké a pri nevyhovení požiadavkám na sto percent odchádza zákazník nespokojný, čo má zlý vplyv na následné referencie pre firmu. Zamestnanec často ani nie je schopný zapamätať si celý sortiment tovaru a ceny za tovar, čo by umožnilo urýchlene zistiť mobilné zariadenie obsahujúce systém UnIS so zosynchronizovanou databázou tovaru.

Náprava – ucelený systém poskytujúci funkcie každodenne používané a uľahčujúce prácu jednotlivým pozíciám vo firme. Prostredie, v ktorom bude zahrnutá funkcionalita bude pre jednotlivé role intuitívne. Pomôže celej skladovej evidencii, ako aj užívateľom v roli reštauračného predaja, pretože budú okamžite vidieť aktuálne stavy jednotlivých účtov a pri objednávkach môžu využívať prenosné zariadenie, ktoré nahradí doterajšiu papierovú evidenciu.

3.1.5 Stakeholders a užívatelia pre klienta DA

Administrátor:

- **Reprezentuje** chod a manažovanie firemného systému z hľadiska správy firmy a zamestnancov.
- **Zodpovednosti** za inštaláciu systému a správne nastavenie firmy pri inštalácií. Možnosť prestavovať firmu má aj počas celého obdobia používania systému. Okrem iného má plný prístup k jednotlivým zamestnancom, ktorých môže vytvárať, nastavovať im právomoci, či meniť im údaje. A prístup ku všetkým funkciám, ktoré sú poskytované v ostatných režimoch.

Skladník:

- **Reprezentuje** správu a dohľad nad celou skladovou evidenciou, umiestnením skladov a ich rozdelením do skupín, ako aj tvorbou skladových kariet s nastavením prepočtov pri predaji.

- **Zodpovednosti** za správu skladov si vyžadujú neustálu kontrolu a dohľad nad tovarom, má za úlohu udržiavať položky v skladových skupinách v aktuálnom stave. To znamená kontrolu skladových položiek a skladového množstva, tvorbu vyraďovacích dokladov v prípade poškodenia, alebo inej straty hodnoty. Vytvárané skladové karty musia byť už upravené na predaj a to nastavením prepočtového koeficientu. Táto rola zahrňuje aj napĺňanie skladov položkami na základe vytvárania príjmových dokladov s dodávateľmi a registrovanie či úpravu stávajúcich dodávateľov a vytváranie tlačových výstupov v prípade potreby. Navyše má zodpovednosť aj za predajné funkcie, keďže má možnosť vytvárať výdavkové doklady na odberateľov.

Predajca:

- **Reprezentuje** vytváranie účtov pre zákazníkov a to v pozícií reštauračného predaja, teda výdajové doklady bez odberateľov. Alebo v pozícií jednotlivého predaja odberateľom.

- **Zodpovednosti**, ktoré obsahuje táto pozícia priamo vplývajú na potenciálnych zákazníkov. Jedná sa o priamy kontakt systému s predajcom a zákazníkom v podobe vytváraného a vytvoreného účtu, či výdajky na odberateľa. Predajca má dostupnú funkciu elektronickej peňaženky, ktorá slúži na jeho vnútornú kontrolu a má zodpovednosť za výbery a vklady. Každú operáciu v elektronickej peňaženke musí odôvodniť a tá je zapisovaná do externého eXtensible Markup Language (XML) súboru s dátumom a časom.

Manažér:

- **Reprezentuje** vedenie obchodu z manažérskeho hľadiska.

- **Zodpovednosti** manažéra nie sú úzko spojené so systémom, pretože má za úlohu dohľad nad dodržiavaním zásad určených firmou, určovanie nových trendov a prijímanie personálu výberovými konaniami.

Možný zákazník:

- **Reprezentuje** zákazníka v pozícií objednávateľa tovaru a následného platcu, ktorému systém vystavuje potvrdenie o úhrade požadovaného tovaru, ktorý je dostupný.

- **Zodpovednosti** zákazníka nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme uhradiť účet.

Dodávateľ:

- **Reprezentuje** dodávateľa v pozícií zásobovania firmy objednaným, alebo zakúpeným tovarom.

- **Zodpovednosti** dodávateľa nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme dodať objednaný tovar.

Odberateľ:

- **Reprezentuje** odberateľa, ktorému je vystavovaný daňový doklad na zvolený tovar a následné zaplatenie.

- **Zodpovednosti** odberateľa nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme zaplatiť účet.

3.1.6 Stakeholders a užívatelia pre klienta MA

Prihlásený užívateľ:

- **Reprezentuje** vytváranie, vymazávanie, upravovanie a zobrazovanie účtov pre zákazníkov.

- **Zodpovednosti** za synchronizáciu údajov z databázy mobilného zariadenia do centrálnej databázy firmy a za správne nastavenie zariadenia pre komunikáciu s klientom. Táto rola má prístup do systému, preto na nej záleží v akom stave bude udržiavať databázu mobilného zariadenia, teda či bude synchronizované účty pravidelne premazávať, pretože by mohlo hroziť preplnenie pamäte.

Neprihlásený užívateľ:

- **Reprezentuje** nastavenie pripojenia mobilného klienta k serveru a synchronizáciu užívateľov registrovaných v centrálnej databáze.

- **Zodpovednosti** za správne nastavenie zariadenia pre komunikáciu so serverom a synchronizáciu registrovaných zákazníkov s centrálnou databázou UnIS(u), aby sa mohol prihlásiť do systému.

Manažér:

- **Reprezentuje** vedenie obchodu z manažérskeho hľadiska.

- **Zodpovednosti** manažéra nie sú úzko spojené so systémom, pretože má za úlohu dohľad nad dodržiavaním zásad určených firmou, určovanie nových trendov a prijímanie personálu výberovými konaniami.

Možný zákazník:

- **Reprezentuje** zákazníka v pozícií žiadateľa tovaru a následného zaplatenia.

- **Zodpovednosti** zákazníka nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme zaplatiť účet.

3.1.7 Užívateľské prostredie

Užívateľské prostredie by malo byť intuitívne a jednoducho ovládateľné. Malo by byť schopné komplexnej evidencie tovaru a pokryť všetky potreby skladovania, ako aj evidenciu externých skladov na iných adresách ako je adresa sídla firmy.

Pokiaľ sa jedná o klienta MV, potom by malo byť ovládanie prispôsobené na minimálne používanie vyvolávacej klávesnice, teda vytvorenie aplikácie ovládanej prevažne klikaním na display.

Zadávateľ si vyžaduje rozšírenia ako:

- možnosti úpravy prepočtu ceny tovaru pri predaji,
- možnosti vymazávania výdajok a následné vrátenie stavu skladu,
- možnosti vymazávania stornovacích dokladov s vrátením stavu skladu,
- možnosti upravovať ceny nákupu na skladových kartách pri vytváraní príjemiek.

Zadávateľ si vyžaduje rozšírenia pre mobilné zariadenia ako:

- Zmenu počtu u položiek za pomoci + a – tlačidiel, ktoré vždy upravia hodnotu o predávanú jednotku.
- Zobrazovanie predajnej ceny a jednotky v sklade tovaru.
- Ponechanie zaplateného účtu v databáze mobilného zariadenia po synchronizácií s možnosťou neskoršieho vymazania.

3.1.8 Stakeholders a užívateľské profily pre klienta DA

Administrátor:

- **Popis** – priamy užívateľ systému,
- **Predpokladané schopnosti** – znalosť architektúry firmy a hierarchie pridelenia práv zamestnancom. Schopnosť pohybovať sa v systéme a vedieť využívať všetky funkcie,
- **Kritéria úspechu** – nainštalovanie systému a vytvorenie údajov o firme, registrácia a pridelenie práv zamestnancom, orientácie v oblasti skladu a predaja systému.

Skladník:

- **Popis** – priamy užívateľ systému,
- **Predpokladané schopnosti** – znalosť skladového hospodárstva danej firmy a schopnosť pohybovať sa v systéme a vedieť využívať všetky funkcie týkajúce sa skladovej evidencie,
- **Kritéria úspechu** – vytváranie a správa umiestnenia jednotlivých skladov a skladových skupín, vytváranie skladových kariet a nastavovanie prepočtu tovaru pri predaji, správa príjemiek, výdajok, odberateľov a dodávateľov.

Predajca:

- **Popis** – priamy užívateľ systému,
- **Predpokladané schopnosti** – je požadovaná znalosť systému v oblasti predaja, orientácia vo vytvorených skladoch a pri vytváraní viacerých účtov na reštauračnej pokladni,
- **Kritéria úspechu** – rýchle vyhľadanie požadovaného produktu s využitím filtrov a vyhľadávacích kritérií, priradenie vyhľadaného tovaru na účet a jeho úprava v prípade potreby. Zaregistrovanie nových odberateľov a priradenie na výdajky, vystavenie výdajky a vytlačenie zákazníkovi.

Manažér:

- **Popis** – nie je priamy užívateľ systému,
- **Predpokladané schopnosti** – komunikatívnosť, výstrednosť a pochopenie požiadaviek zákazníkov v osobnom sektore,
- **Kritéria úspechu –** fungovanie firmy podľa predstáv majiteľa a vedenia. Teda, keď sú spokojní zákazníci aj zamestnanci.

Možný zákazník:

- **Popis** – nie je priamym užívateľom systému,
- **Predpokladané schopnosti** – zákazníka nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme zaplatiť účet,
- **Kritéria úspechu** – spokojnosť so službami firmy, obsluhou, efektivitou a tlačovými výstupmi.

Dodávateľ:

- **Popis** – nie je priamy užívateľ systému,
- **Predpokladané schopnosti** – spoľahlivosť a splnenie kvality produktov poskytovaných firme,
- **Kritéria úspechu** – jednoznačným úspechom je, keď tovar sľúbený a objednaný firmou, je dodávateľom včas a v dohodnutom množstve dopravený.

Odberateľ:

- **Popis** – nie je priamy užívateľ systému,
- **Predpokladané schopnosti** odberateľa, rovnako ako možného zákazníka nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme zaplatiť účet,
- **Kritéria úspechu –** spokojnosť pri výbere tovaru, jeho dostupnosti, s obsluhou, efektivitou a tlačovými výstupmi.

3.1.9 Stakeholders a užívateľské profily pre klienta MA

Prihlásený užívateľ:

- **Popis** – priamy užívateľ systému,
- **Predpokladané schopnosti** – znalosť predaja a vytváranie účtov zákazníkom a schopnosť pohybovať sa v systéme. Tiež orientácie v skladoch pre rýchle vyhľadávanie položiek,
- **Kritéria úspechu** – vytvorenie a správa účtov, synchronizácia databázy mobilného zariadenia s centrálnou databázou a nastavenie spojenia k serveru.

Neprihlásený užívateľ:

- **Popis** – priamy užívateľ systému,
- **Predpokladané schopnosti** – znalosť sieťového zabezpečenia a schopnosť nastavenia aplikácie pre pripojenie k serveru. Tiež synchronizácia registrovaných zamestnancov z centrálnej databázy do mobilného zariadenia,
- **Kritéria úspechu** – nadviazanie pripojenia na server a synchronizácia užívateľov.

Manažér:

- **Popis** – nie je priamy užívateľ systému,
- **Predpokladané schopnosti** – komunikatívnosť, výstrednosť a pochopenie požiadaviek zákazníkov v osobnom sektore,
- **Kritéria úspechu –** fungovanie firmy podľa predstáv majiteľa a vedenia. Keď sú spokojní zákazníci aj zamestnanci.

Možný zákazník:

- **Popis** – nie je priamym užívateľom systému,
- **Predpokladané schopnosti** – zákazníka nie sú špecifikované, pretože nemá priamy kontakt so systémom, len záväzok firme zaplatiť účet,
- **Kritéria úspechu** – spokojnosť so službami firmy, obsluhou, efektivitou a pohotovosťou zamestnancov vďaka UnISu na mobilných zariadeniach.

3.1.10 Nároky na kvalitu

Táto časť definuje nároky na výkon, robustnosť, odolnosť, použiteľnosť, a podobné charakteristické rysy systému UnIS.

Dostupnosť – systém bude k dispozícií 24 hodín denne, 7 dní v týždni.

Použiteľnosť – systém bude ľahko (intuitívne) použiteľný a bude určený zaškoleným užívateľom do problematiky obchodu.

Použiteľnosť – systém bude zahŕňať pomoc pre užívateľov. Bude sa jednať o nápovedu programu v stavovom riadku spusteného hlavného okna. Prípadne doložené programátorské príručky.

Udržiavateľnosť – systém bude určený pre ľahké technické zabezpečenie. Všetky dáta v systéme budú zálohované pravidelne s príslušnými bodmi obnovy pri páde systému. Záloha je nastavovaná pri inštalácií databázy na databázový server. Prednostne MS SQL Server. U mobilných zariadení bude vlastnú databázu tvoriť prednostne Microsoft SQL CE.

3.2 UML

Pre jednoznačnejšie spracovanie špecifikácie, analýz a návrhu je použitý UML(Unified Modeling Language). Jedná sa o grafický jazyk určený na vizualizáciu, špecifikáciu, navrhovanie a dokumentáciu programových systémov a je určený predovšetkým pre softvérové inžinierstvo.

Pomocou UML môžete vytvárať štandardne zápisy alebo návrhy systémov vrátane konceptuálnych prvkov, ako sú business procesy a systémové funkcie, tak konkrétnych prvkov ako sú príkazy programovacieho jazyka, databázové schémy a znovu použiteľné programové komponenty. Hlavným cieľom UML je podpora objektovo orientovaného prístupu pri vytváraní analýz a návrhov programových systémov. Štandard UML definuje štandardizačná skupina Object Management Group (OMG) [7].

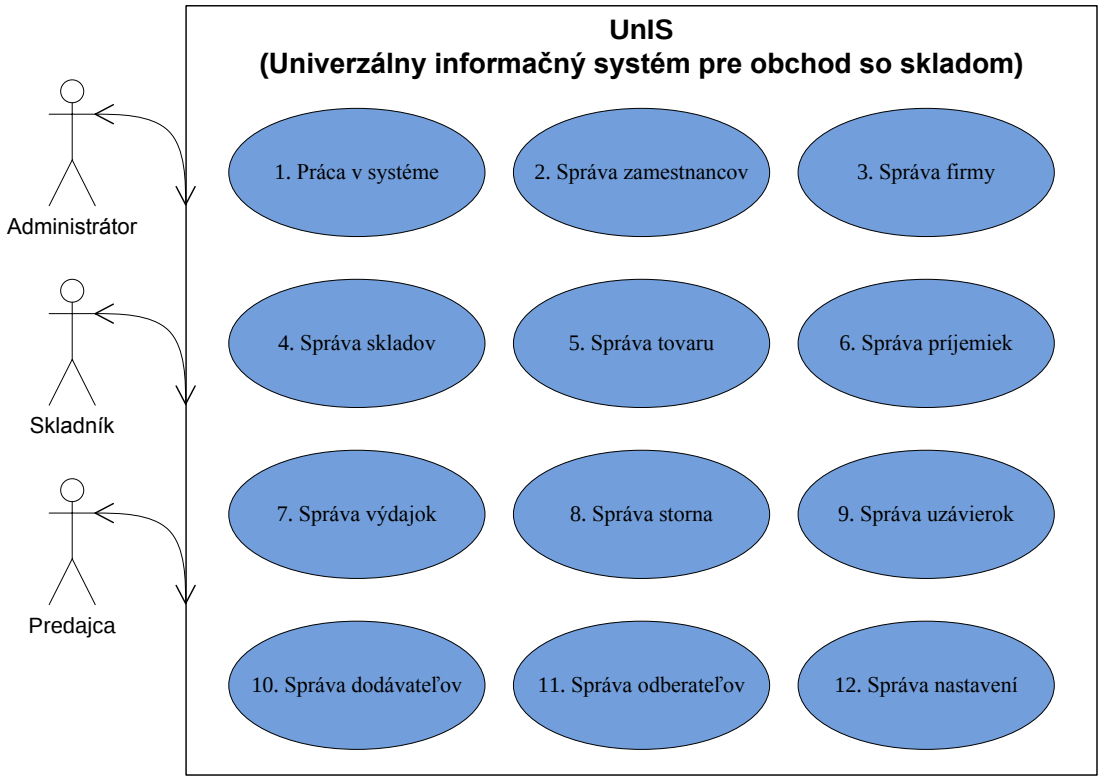
3.3 Funkčný náhľad

V tomto náhľade sme vytvorili kľúčové Use Case diagramy, ktoré vychádzajú zo špecifikovaných požiadaviek na vytváraný systém a zobrazujú funkcionalitu kľúčových funkcií.

Pri vytváraní diagramov, bol použitý nástroj Microsoft Office Visio 2007.

Na vytvorenie funkcií požadovaných k tvorbe systému boli využité pohľady jednotlivých rolí užívateľov, obsiahnuté v kapitole 3.1 Špecifikácia UnIS. Medzi hlavné funkcie zahrnuté v Use Case diagramoch patrí aj klient MA určený pre mobilné zariadenia. Rozdiel oproti klientovi DA, je vo využití funkcií, hlavne kvôli obmedzenému priestoru a ovládaniu mobilných zariadení.

Hlavná funkcionalita UnIS

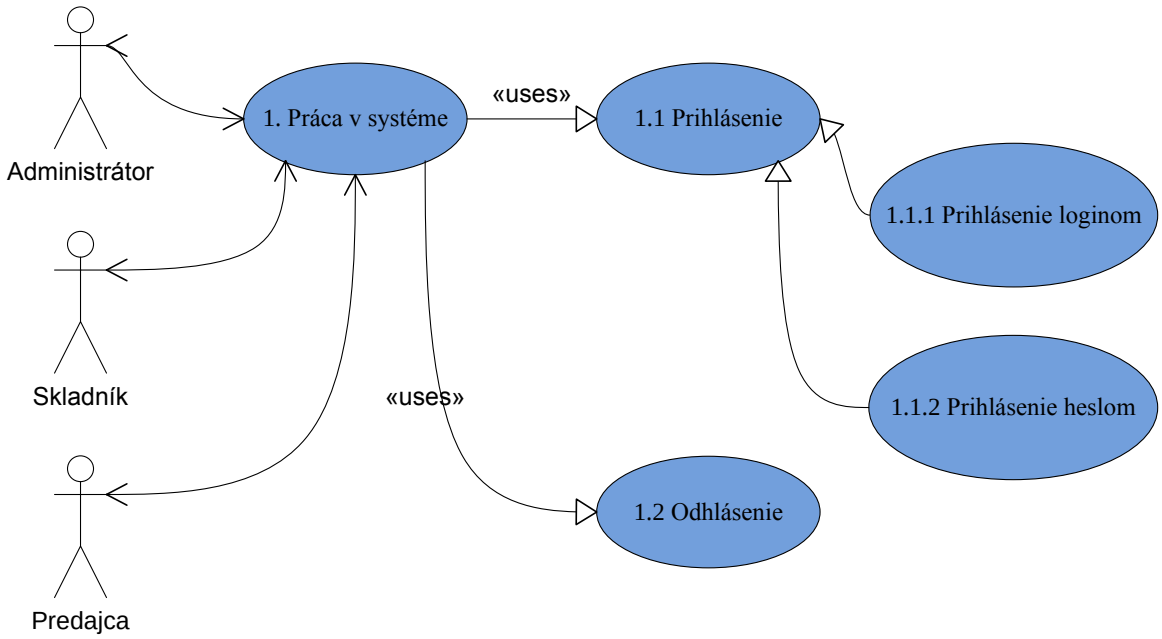


Obrázok 3: Hlavný Use Case UnIS

Práca v systéme

Užívatelia systému sa musia prihlásiť pred vstupom do im určených rozhraní na prácu v systéme. Prihlásenie bude pozostávať z jedinečného zaevidovaného prihlasovacieho mena (login) a hesla, ktoré budú nastavené užívateľovi administrátorom systému UnIS.

Minimálne predpoklady pre prácu v systéme, ako aj prihlásenie do systému sú správne nastavenie pripojenia k webovým službám, ktoré budú poskytovať služby servera a správne nastavený connection string, umožňujúci pripojenie servera k databáze s hierarchiou UnIS.



Obrázok 4: Use Case práca v systéme

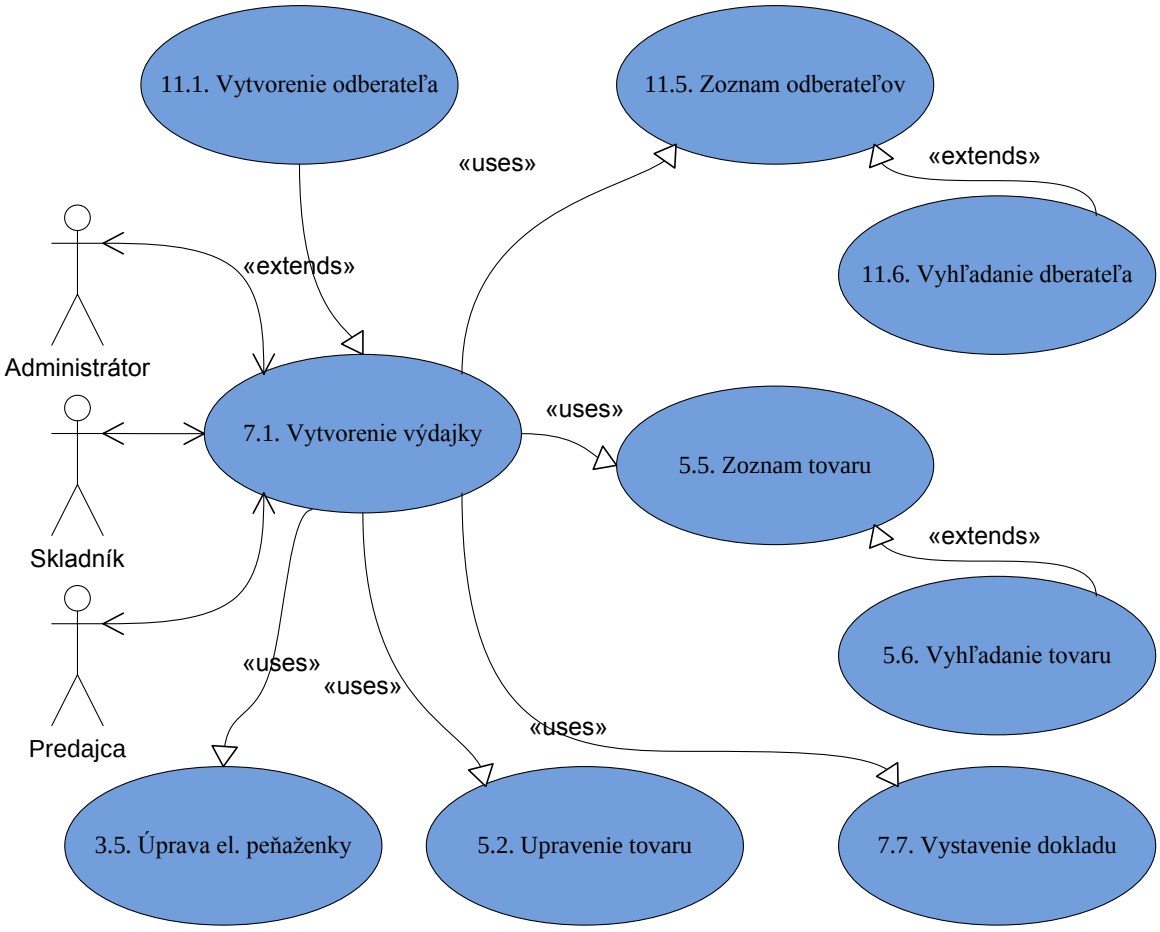
Vytvorenie výdajky

Jedná sa o kľúčovú funkciu pre užívateľa v roli predajcu. Nebude záležať na druhu klienta, táto funkcia bude veľmi dôležitá a využívaná v oboch variantoch. Obmedzenia, ktoré sú viazané k mobilným zariadeniam, už boli spomenuté a z toho dôvodu bude využitie všetkých funkcií vytvárania výdajky obmedzené.

Táto funkcia bude zahrňovať viacero oblastí dôležitých pre jej správny chod. Tu patrí vytváranie odberateľov, ktorému bude výdajka vystavovaná a nebude v databáze evidovaný. Ak odberateľ vytvorený bude, potom možnosť priradenia na doklad zo zoznamu odberateľov. Tiež bude požadovaná spolupráca vytvorenej výdajky s elektronickou peňaženkou, teda zmena ceny

podľa vystavenej objednávky a spolupráca so skladovými kartách. Touto spoluprácou je myslené upravovanie aktuálnych počtov kusov, alebo iných merných jednotiek v sklade pri prevedení platby.

Minimálne predpoklady pri vytváraní výdajok zahrňujú nutnosť užívateľa, aby sa prihlásil do systému UnIS a v prípade klienta pre mobilné zariadenia, aby bola synchronizovaná databáza produktov s centrálnou databázou.



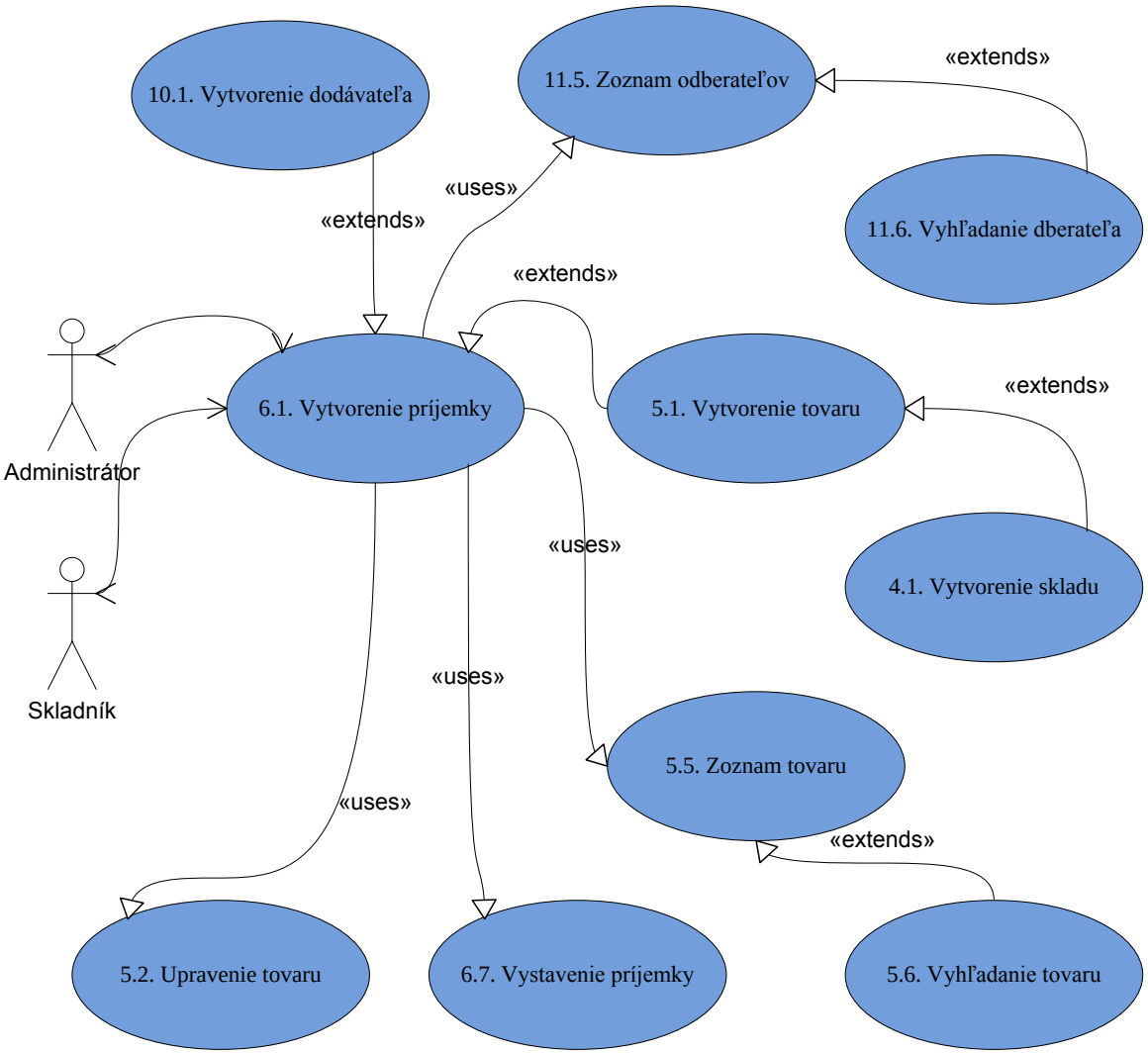
Obrázok 5: Use Case vytvorenie výdajky

Vytvorenie príjemky

Jednou z kľúčových funkcií pre užívateľa v roli skladníka, ktorá však nie je zahrnutá v klientovi DA pre mobilné zariadenia. Dôvodom bola nepotrebnosť klienta tohto druhu a tiež časová zložitosť ovládania pri vytváraní skladových kariet na takomto type klienta.

Rovnako ako ostatné funkcie systému, ani táto nebude výnimkou a bude k svojmu chodu využívať ďalšie moduly. Sem patrí rozšírenie o vytváranie dodávateľov, ktorý nie sú evidovaný centrálnej

databáze, alebo vytvorenie tovaru (skladovej karty), ktorý pri tvorbe príjemky nie je v sklade evidovaný. Rozšírením vytváraného tovaru je aj vytvorenie skladu, ak to bude potrebné. Ďalšou využívanou funkciou bude pridávanie tovaru do príjemky zo zoznamu tovaru, rozšíreného o vyhľadávanie, ako aj vystavenie príjemky, upravenie položky príjemky, alebo úprava stavu tovaru.



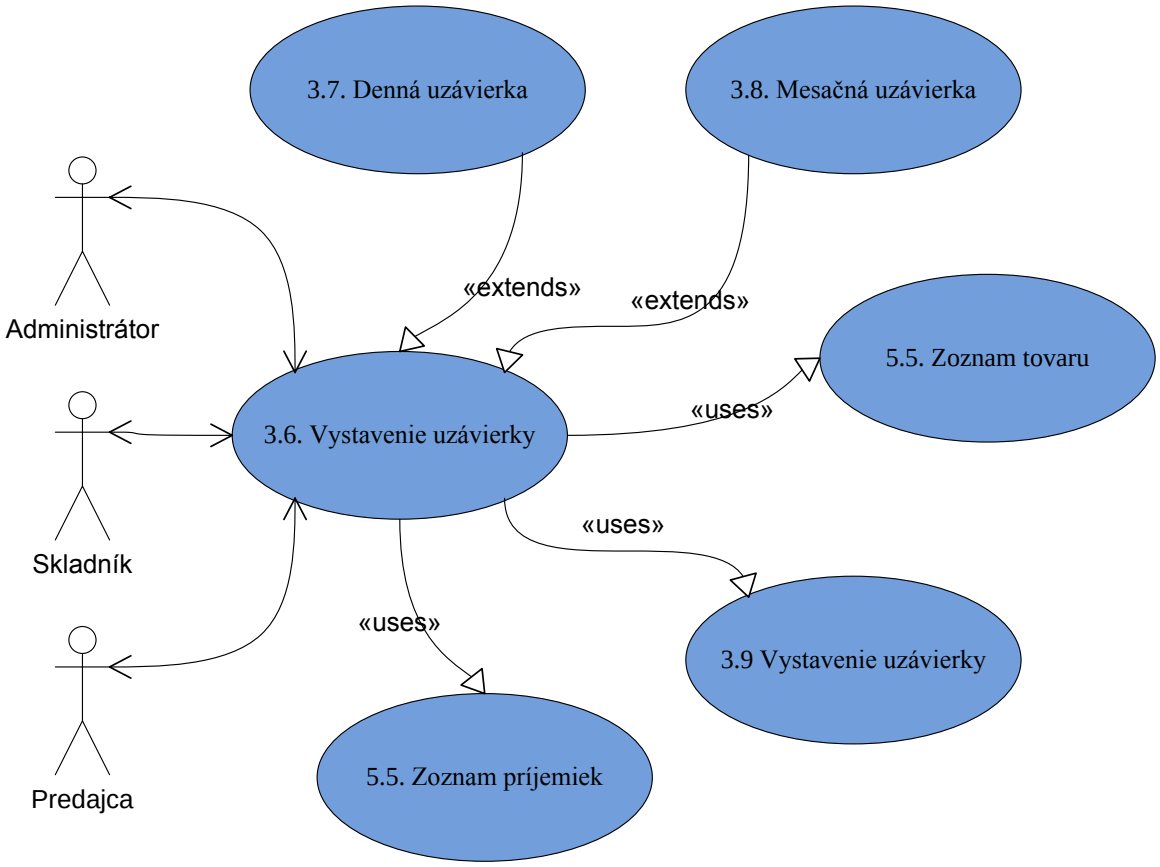
Obrázok 6: Use Case vytvorenie príjemky

Minimálne predpoklady pre vytváranie príjemky zahrňujú nutnosť užívateľa, aby sa prihlásil do systému UnIS.

Vystavenie uzávierky

Veľmi požadovanou funkciou pre roli administrátora je vytváranie uzávierok, aby mal prehľad nad predaným tovarom. Táto funkcia nebude zahrnutá v klientovi MA pre mobilné zariadenia.

Spolupráca tejto funkcie s ostatnými systémovými funkciami bude menšia ako majú ostatné spomínané kľúčové funkcie. Bude využívať dáta zaznamenané vo výdajkách a to hlavne atribúty obsahujúce dátum vytvorenia výdajky. Na základe dátumu bude využívať záznamy položiek príjemky s vyhovujúcim dátumom a z nich získavať potrebné počty o tovare na uzávierku. Následne bude možnosť uzávierku prekonvertovať do PDF súboru a vytlačiť.



Obrázok 7: Use Case vystavenie uzávierky

Minimálne predpoklady pri vytváraní uzávierok, je aspoň jedna výdajka v požadovanom časovom rozsahu a prihlásenie do systému UnIS.

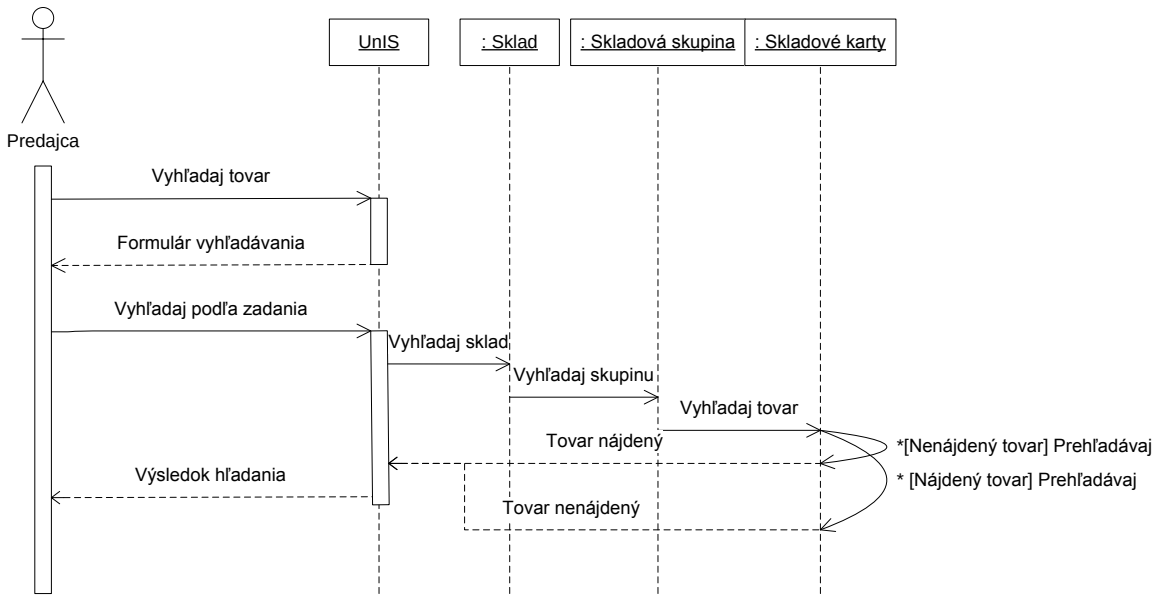
3.4 Dynamický náhľad

Pomocou diagramov UML, reprezentujúcich dynamické chovanie systému, priblížime vzájomnú interakciu objektov a funkcií systému UnIS z časového hľadiska vykonávania. Rovnako ako pre ostatné diagramy bol použitý nástroj Microsoft Office Visio 2007.

Sekvenčné diagramy

Vytváraný softwarový systém nie je tvorený jediným objektom, ale celou radou, medzi sebou komunikujúcich objektov. Touto komunikáciou jednotlivých objektov sa zaisťovaná požadovaná funkcionalita opísaná v kapitole: 3.1 Špecifikácia UnIS. Pre vzájomnú komunikáciu sa využíva prepojenie objektov, čo v konečnom dôsledku tvorí celý systém.

Jazyk UML umožňuje túto komunikáciu zachytiť pomocou sekvenčných diagramov, ktoré zobrazujú komunikáciu medzi objektmi z časového hľadiska.



Obrázok 8: Sekvenčný diagram vyhľadania tovaru

Sekvenčný diagram zobrazuje komunikáciu medzi objektmi systému a tiež komunikáciu systému ako celku s užívateľom, ktorý je v roli predajcu. Ako môžete vidieť, sú znázornené dve situácie, vychádzajúce z funkcie vyhľadávania skladových kariet. Podľa zadaného filtra sa najskôr systém dostane do daného skladu, potom do skladovej skupiny, v ktorej vyhľadáva skladové karty. Užívateľ bude oboznámený s výsledkom spracovania, teda mu bude zobrazený zoznam tovaru, prislúchajúci vyhľadávacím parametrom.

3.5 Analýza a návrh systému

Táto kapitola je venovaná komplexnému staviteľskému pohľadu na vytváraný systém, obsahuje radu rôznych staviteľských pohľadov pre priblíženie problému. Cieľom je zachytiť a sprostredkovať významné vývojové rozhodnutia ktoré budú využívane pri budovaní systému.

Návrhy vytvorené v časti analýzy musia splňovať nasledujúce obmedzenia:

- Administrátor bude mať v systéme najvyššiu právomoc a musí mať prístup k všetkým funkciám, ktoré mal dostupné pred vytvorením a nasadením systému. To znamená, že systém nesmie znižovať, alebo inač obmedzovať jeho postavenie vo firme.

- Navrhnuté rozhranie bude musieť byť prispôsobené jednotlivým rolám užívateľov systému a po konzultácií s kľúčovými užívateľmi prispôsobené pre urýchlenie a uľahčenie práce.

- Vytvorený klienti budú mať prístup k centrálnej databáze až po prihlásení užívateľov. Prihlásený užívatelia budú môcť využívať funkcie systému, ktoré budú zodpovedať ich postaveniu vo firme na základe priradených roli pri registrácií.

- Aplikácia bude obsahovať klientsku časť, ktorá bude rozdelená na klienta DA ako desktopová aplikácia a klienta MA ako mobilná aplikácia. Klient MA bude obsahovať vlastnú databázu, ktorú bude synchronizovať s centrálnou a klient DA bude fungovať ako tenký klient.

- Server bude umožňovať využívanie funkcií na prístup a správu centrálnej databázy UnIS.

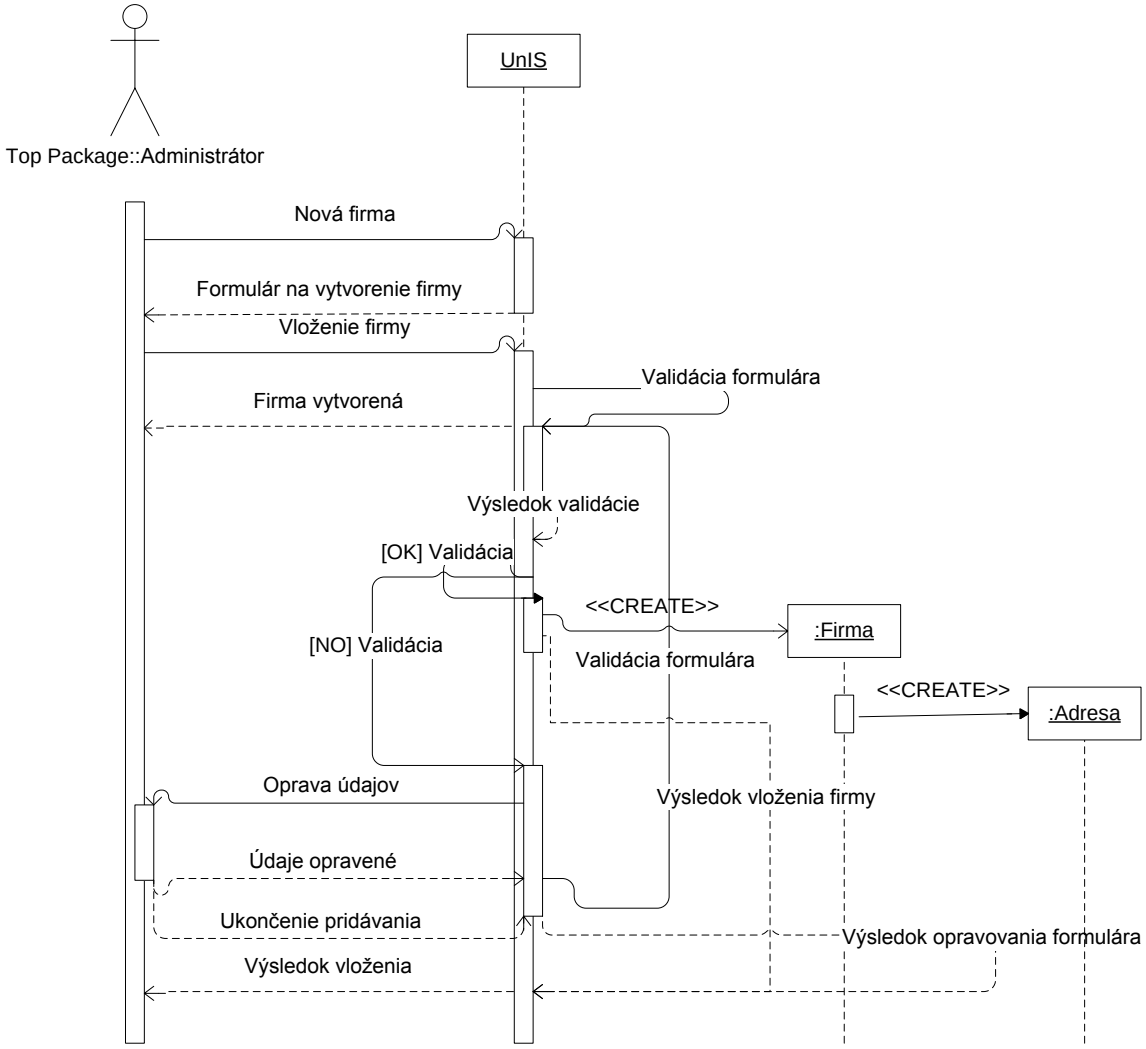
Hierarchia celého systému aj s diagramom nasadenia je popísaná nižšie. Niektoré ikony použité v klientskej časti aplikácie budú použité z webovej adresy: http://www.kasman.sk/ikonky.

3.5.1 Dynamický model

Sekvenčný diagram

Sekvenčné diagramy použité v predchádzajúcej kapitole: 3.4 Dynamický náhľad sa používajú aj v návrhu, pretože umožňujú lepšie popísať interakciu medzi objektmi z hľadiska časového usporiadania.

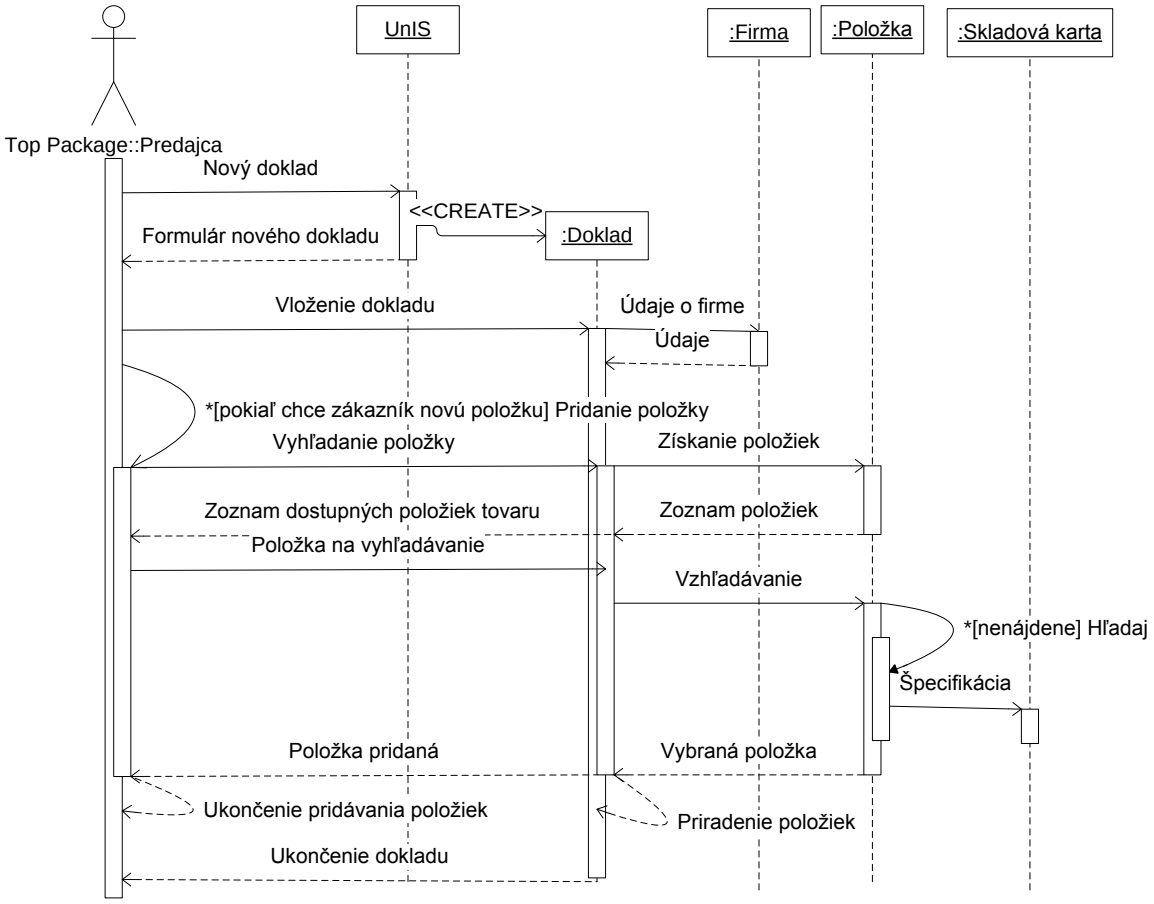
Nasledujúci obrázok na ktorom je sekvenčný diagram, znázorňuje interakciu systému s objektmi a komunikáciu medzi užívateľom a systémom, ktorý poskytuje formulár na vloženie firmy a po jeho vyplnení užívateľom a zaslaní späť do systému skontroluje jeho správne vyplnenie. Ak dôjde k nesprávnemu vyplneniu, zašle užívateľovi správu o nesprávnom vyplnení a ten môže formulár opraviť, alebo ukončiť vkladanie. Ak opraví formulár, systém prejde znova k validácií a následne vytvorí objekt firma s vyplnenými atribútmi, ktorá vytvorí objekt adresa k nej prislúchajúci. To však len za predpokladu, že bude validácia formulára správna.



Obrázok 9: Sekvenčný diagram vytvorenia firmy

Za najdôležitejší sekvenčný diagram, ktorý tvorí jadro celého predaja tovaru zo skladov a pracovanie so skladom či skladovými kartami, sme vybrali proces vytvárania dokladu. Pričom sme neprihliadali na všetky možnosti, ktoré môžu behom vytvárania nastať.

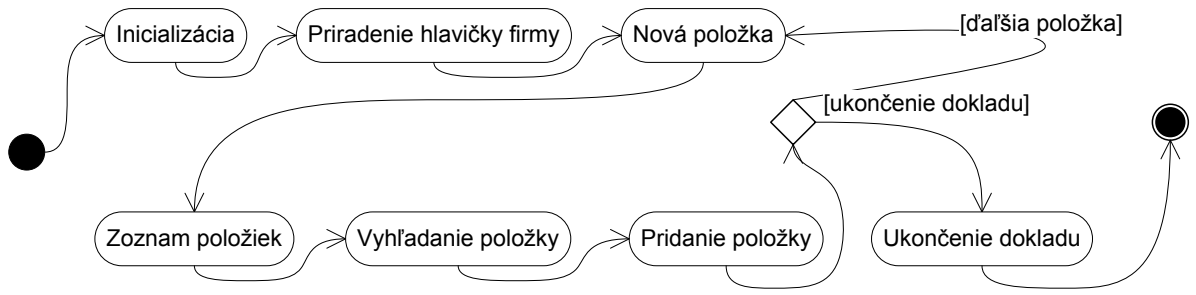
Nasledujúci obrázok sekvenčného diagramu pre vytváranie dokladu charakterizuje postupnosť krokov, kde sme zahrnuli vyhľadávanie a pridávanie položiek na doklad. Užívateľ systému môže podľa zadávania zákazníkom pridať ľubovoľný počet položiek na vystavovaný doklad. Pre prehľadnosť neboli do diagramu zahrnuté možnosti pridelenia odberateľa a v prípade, že odberateľ neexistuje, pridanie nového záznamu o odberateľovi, ktorý by potom mohol byť na doklad pridaný.



Obrázok 10: Sekvenčný diagram vytvorenia dokladu

Stavový diagram

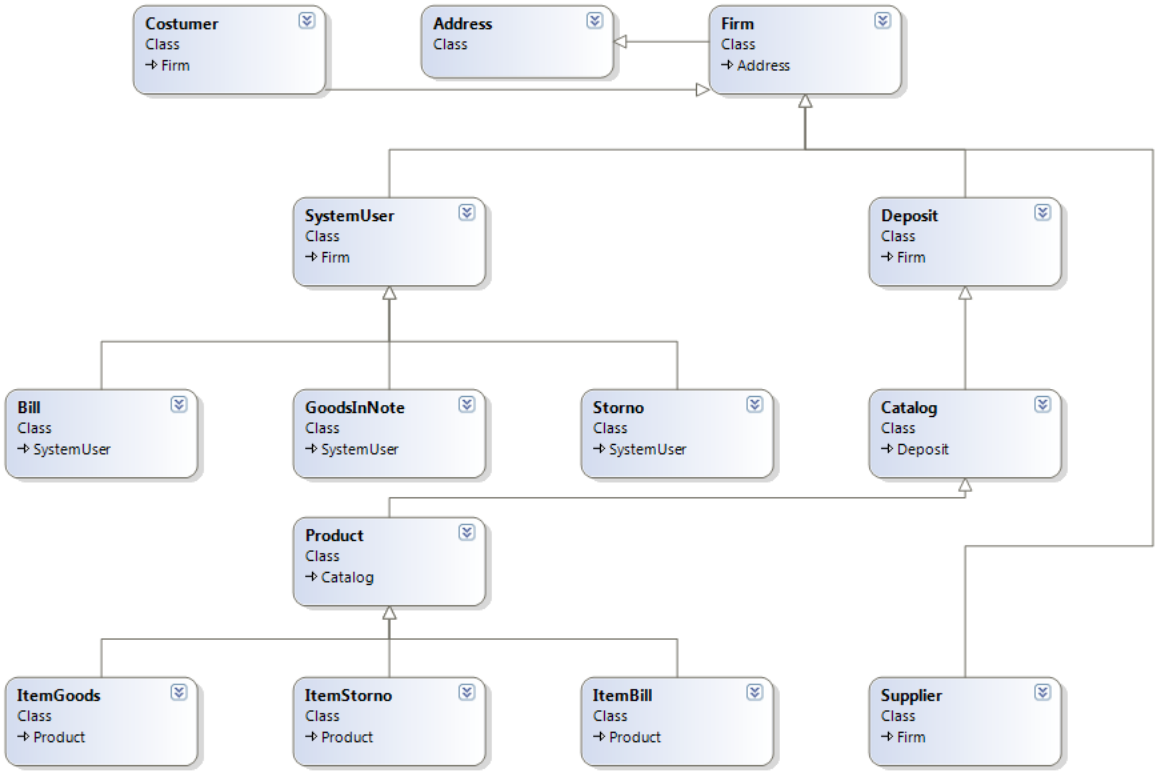
Priblíženie životného cyklu objektu, zobrazíme pomocou stavového diagramu vychádzajúceho zo sekvenčného diagramu pre vytvorenie objednávky je znázornené na obrázku: Obrázok 11: Stavový diagram pre objekt - Doklad obrázok charakterizuje stavy objektu, udalosti ktoré spôsobujú zmeny týchto stavov a nakoniec akcie, ktoré z týchto zmien vyplývajú.



Obrázok 11: Stavový diagram pre objekt - Doklad

3.5.2 Statický model

K vyjadrenie statického pohľadu na vytváraný systém slúžia diagramy tried jazyka UML, ktorý je zobrazený na nasledujúcom obrázku.



Obrázok 12: Diagram tried UnIS

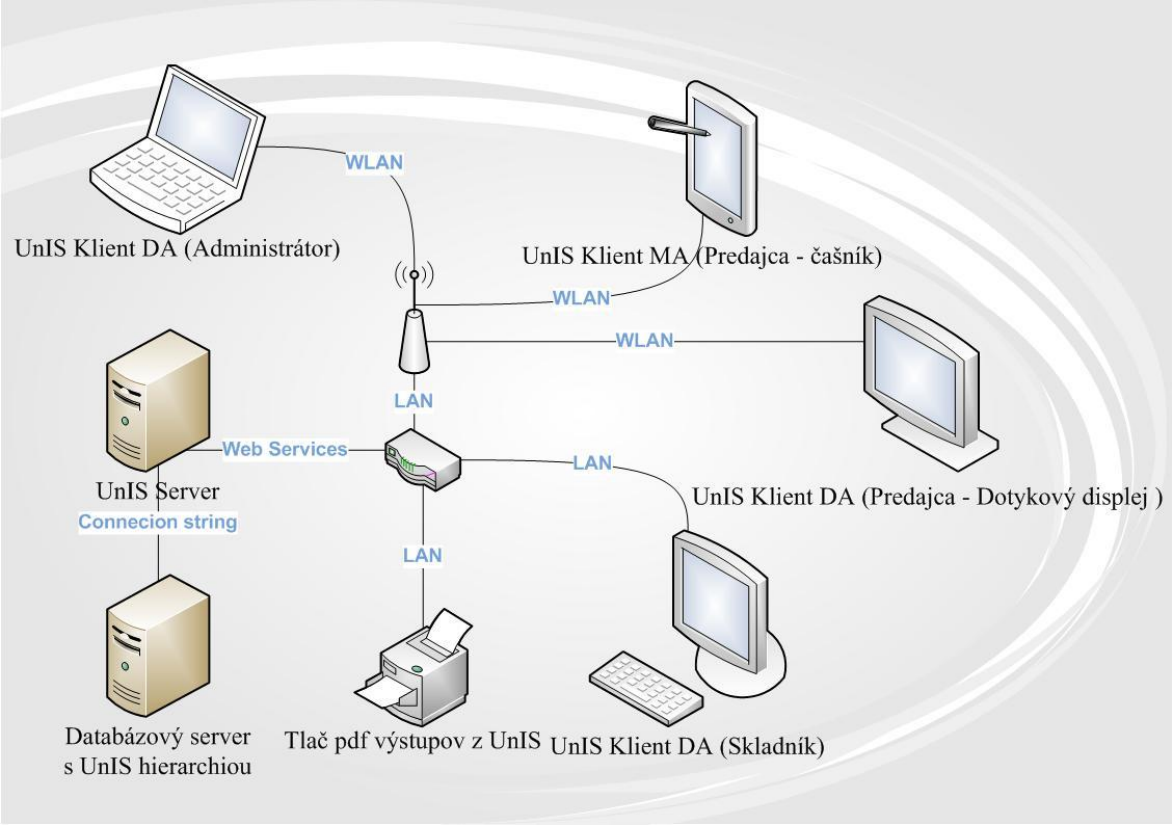
3.5.3 Model nasadenia systému UnIS

Nasledujúci obrázok opisuje nasadenie systému UnIS a jednotlivé role, alebo rozhrania poskytované systémom. Bližšie si popíšeme ako má byť systém využívaný vo firme a opíšeme komunikáciu

jednotlivých časti systému. Sem patrí komunikácia klientov so serverom a komunikácia servera s centrálnou databázou.

UnIS Klient DA (Administrátor) bude využívaný administrátorom a pripojený na firemnú sieť wifi, pomocou ktorej sa dostane k webovým službám, poskytovaným serverom. Touto cestou mu bude umožnené komunikovať so serverom a spravovať tak všeobecné nastavenia servera pre prístup k databáze ako aj správu všetkých funkcií poskytovaných serverom.

UnIS Klient MA (Predajca „čašník“), ktorého prevažným užívateľom bude predajca „čašník“, bude slúžiť na využívanie pokladnice pre mobilné zariadenia. Komunikáciu so serverom zaistí pripojenie na firemní wifi sieť, alebo na internet prostredníctvom wifi. Internetové pripojenie bude potrebné len v prípade umiestnenia servera mimo prevádzky, tak by boli jeho služby dostupné cez internet.



Obrázok 13: Nasadenie UnIS

UnIS Klient DA (Predajca – dotykový displej) predstavuje nasadenie klienta na počítač v predajni, alebo reštaurácií s možnosťou využitia dotykového displeja. Jeho úlohou bude vytváranie reštauračných dokladov a evidencia odberateľov spolu s výdajkami do centrálnej databázy na serveri. Preto musí byť zaistená komunikácia so serverom, už spomínanými spôsobmi.

UnIS Klient DA (Skladník) opisuje rolu skladníka, využívanú na počítači umiestnenom v skade a umožňujúcom udržiavať a spravovať skladové hospodárstvo firmy. Predpokladá sa priame napojenie PC pomocou sieťového kábla k serveru, alebo k internetu ak bude server umiestnený mimo firemnú sieť.

Tlač pdf výstupov z UnIS bude reprezentovať tlačiareň zapojená do siete, tak aby ju mohli jednotlivý klienti DA využívať a vytláčať vyhotovené tlačové výstupy do formátu pdf. Možnosť vytláčať budú mať užívatelia aj priamo zo systému bez nutnosti vytvorenia pdf dokumentu, teda z náhľadu vygenerovaného dokumentu.

UnIS server bude predstavovať základný kameň celého systému a poskytovať klientom vyžadovanú funkcionalitu prostredníctvom webových služieb. Jeho prioritnou úlohou bude vytvorenie spojenia s databázou a pomocou SQL príkazov dostať požadované dáta, alebo zmeniť, vytvoriť či zmazať údaje z centrálnej databázy.

Databázový server s UnIS hierarchiou bude obsahovať jadro aplikácie, teda centrálnu databázu s uloženými záznamami. Podľa výberu servera si užívatelia budú môcť nastaviť vytváranie bodov obnovy a zálohovanie, aby sa uistili, že v prípade zlyhania o ne neprídu.

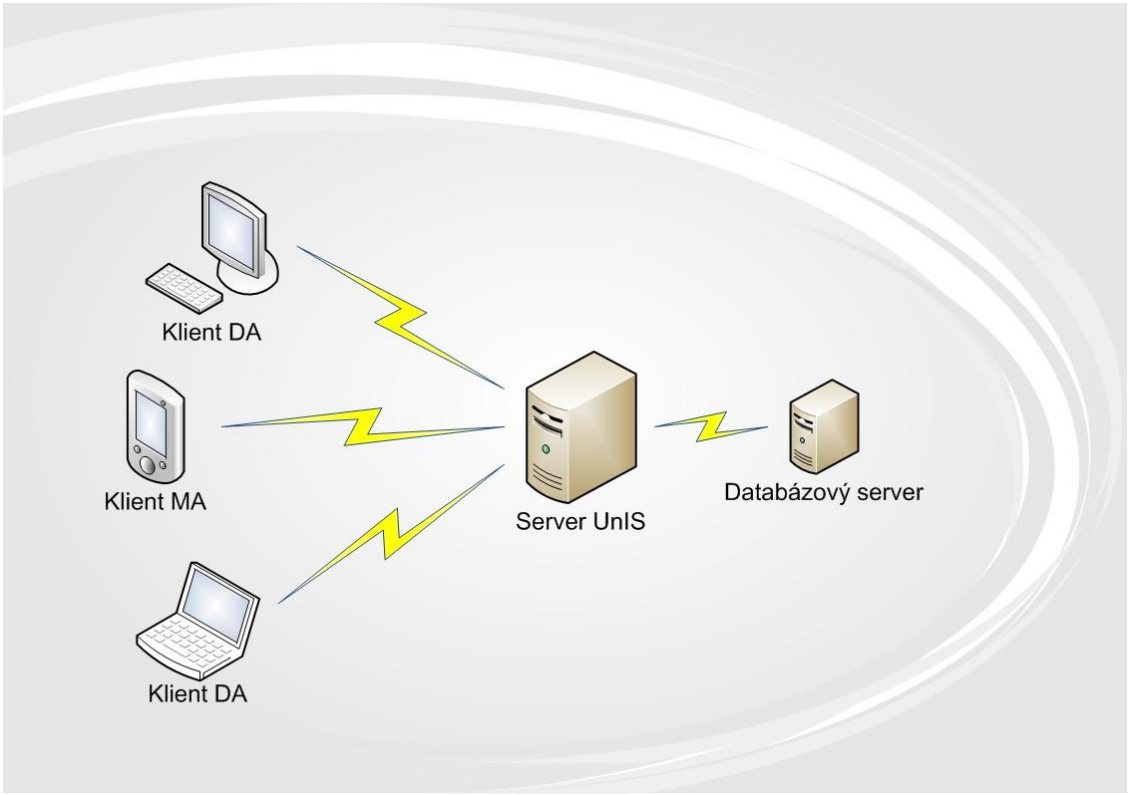
Možnosti nasadenia UnIS systému je oveľa viac, no bol popísaný prípad pre ktorý bol prioritne informačný systém vytvorený. Keďže je zaistená komunikácia pomocou webových služieb systém tak nemá viazaný server na operačnom systéme a vytvárané knižnice na obsluhu databázy s príponou .DLL sú prenositeľné.

4 Realizácia aplikácie UnIS

Fenoménom dnešného programovania sa stáva práve zvolený jazyk na implementáciu nášho projektu UnIS. Je ním C# (vyslovuje sa „si-šarp“), jednoduchý, moderný, objektovo orientovaný jazyk vyvinutý spoločnosťou Microsoft. Jazyky, z ktorých vychádza sú C++ a Java. Jeho prvoradým cieľom je umožniť rýchle programovanie, ktoré už umožňujú jazyky ako Visual Basic (VB), alebo Delphi [4].

Samotná realizácia systému UnIS je rozdelená do troch hlavných častí. Prvou je implementácia klienta DA ako desktopovej aplikácie s využitím návrhového vzoru MVC. Nasledujúcou časťou je vytvorenie serverovej časti, ktorá sa bude starať o prístup do jadra systému, teda centrálnej databázy celého systému. Poslednou časťou je vytvorenie klienta MA pre mobilné zariadenia, ktorý zmodernizuje používanie a zvýši komfort užívateľom, ale rovnako aj zákazníkom, ktorí sa stanú indikátorom kvality poskytovaných služieb firmy, využívajúcej systém UnIS.

Ako je zrejmé, architektúra celého systému je založená na modely klient-server. Týmto modelom je myslené oddelenie systému klienta od servera a vzájomná komunikácia cez počítačovú sieť, pričom softvér klienta sa stará o komunikáciu so serverom a odosielanie požiadaviek na server, ktorý následne vyhodnocuje požiadavky a zasiela klientovi požadované informácie späť.



Obrázok 14: Klient - server architektúra UnIS

Táto komunikácia medzi klientom a serverom je možná aj na jednom počítači, oddelenie klientov od servera a komunikácia prostredníctvom počítačových sietí je veľkou výhodou pri rozmiestňovaní aplikácie na rôzne miesta, s určitým centrom hlavných informácií (v našom prípade to je hlavná centrálna databáza). Pri využití webových služieb a komunikácií po internete je klient takmer úplne voľný z pohľadu nasadenia na rôznych miestach, pričom jeho podmienkou je pripojenie na internet.

Popis klienta

Hlavnou úlohou je naviazanie spojenia so serverom a odosielanie požiadaviek, po odoslaní sa jeho úloha mení na príjemcu, teda očakáva odpoveď od servera. Jedná sa o priamy kontakt systému s užívateľom cez grafické užívateľské rozhranie.

Popis servera

Server je prijímateľ požiadaviek zaslaných klientom a má za úlohu spracovať požiadavku, na ktorú v zapätí odpovedá klientovi. Typické je pripojenie viacerých klientov na jeden server a ten nie je priamo ovplyvňovaný koncovým užívateľom [23].

Na samotnú implementáciu bol použitý vývojový nástroj spoločnosti Microsoft, a to Microsoft Visual Studio 2008. Využívaný bol .NET Framework a .NET Compact Framework pre implementáciu a testovanie jednotlivých zložiek systému. Použitým databázovým serverom bol Microfost SQL Server 2005 s využitím SQL Server Management Studia. Ako databázový server pre vývoj a test mobilného klienta bol využitý Microsoft SQL Server Compact (SQL CE) na operačnom systéme Windows Mobile 6.0.

4.1 Implementácia klienta DA

Pri realizácií klienta ako desktopovej aplikácie, sme použili návrhový vzor Model-View-Controler (MVC). MVC je v poslednej dobe veľmi často zmieňovaný návrhový vzor a tvorí spoľahlivý základ pre kvalitné objektové programovanie. Tvorcom modelu bol známy Trygve Reenskaug, ktorý predstavil tento model pre jazyk Smalltalk, okolo roku 1978. Od tohto dátumu bola snaha tento model zaimponovať do všetkých možných platforiem, ktoré majú niečo spoločné s vytváraním grafického užívateľského rozhrania.

Popularita architektúry MVC má v posledných rokoch dramatický nárast aj kvôli zakomponovaniu modelu do prostriedkov potrebných k tvorbe aplikácii s kvalitnou architektúrou. Motiváciou zaradenia tohto modelu bol aj takzvaný drag & drop vývoj aplikácií, ktorý prináša na jednej strane veľké nadšenie a uľahčenie práce, no na druhej strane predstavuje veľkého strašiaka pri vyhľadávaní chýb, testovaní a hlavne udržiavaní aplikácií. Fáza správy a údržby býva spravidla veľmi nákladná na čas aj financie, preto bolo potrebné neprehľadný kód hierarchizovať a rozčleniť na viacero vrstiev,

ktoré by spolu komunikovali. Práve týmto smerom sa uberá architektúra MVC, ktorá je členená na 3 logické časti, ktoré sú samostatne upravovateľné.

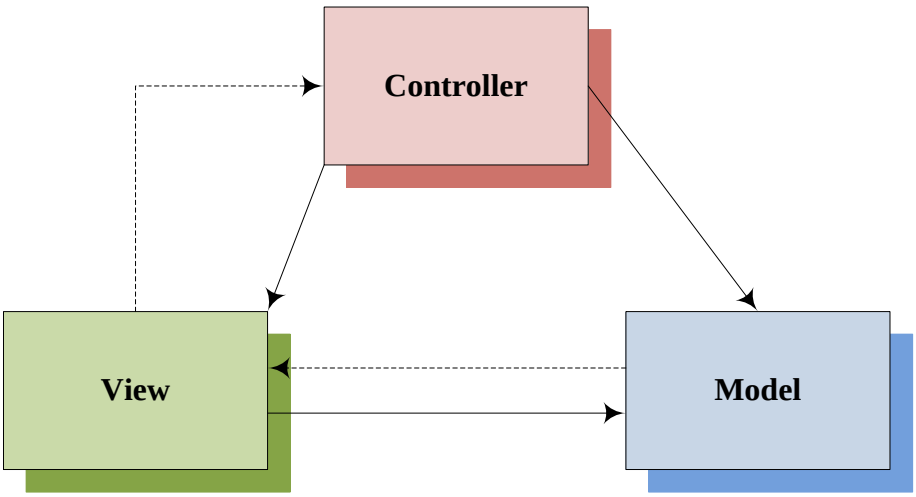
Časti MVC architektúry sa členia na:

- **Model** – reprezentuje hlavnú doménovú logiku aplikácie, to zahŕňa dáta, nad ktorými aplikácia pracuje. Tu patrí databáza, obrázok, textový dokument a mnoho ďalších častí systému. Jedná sa o nevizuálnu časť obsahujúcu spomínané dáta a správanie celej aplikácie, okrem toho, čo sa týka užívateľského rozhrania,

- **View** – je práve ten pohľad na model, ktorý zaisťuje grafické užívateľské rozhranie a umožňuje ho vytvárať nezávisle na ostatných modeloch. To je výhodou pre väčšie projekty, kde užívateľské rozhrania navrhujú grafici a o logickú časť sa starajú programátori. Základom je hlavne to, že táto časť architektúry neobsahuje žiadne vlastné dáta a logiku, o ktoré sa starajú ostatné moduly architektúry, teda len zobrazuje dáta z modelu užívateľovi,

- **Controller** – je takzvanou spojkou medzi modelom a view modulom. To znamená zaistenie logiky a správania celej aplikácie. Funkcia sa dá pochopiť ako príjem udalostí vyvolaných vstupom od užívateľov prostredníctvom view (napríklad stlačenie tlačidla) a zareagovaním na túto výzvu upravením výstupných objektov v rámci view (napríklad zmena farby pozadia). Je tu však potrebná komunikácia controlleru s modelovou časťou architektúry, kde sú údaje získavané z modelu, ktoré si controller vyžiada.

Popísaná architektúra je znázornená na obrázku:



Obrázok 15: Model-View-Controller

Ak vezmeme do úvahy komunikáciu užívateľa s týmto modelom, potom výstup aplikácie bude mať na starosť view. Pokiaľ užívateľ do modelu chce vstúpiť, má na to dve odlišné možnosti. Prvá

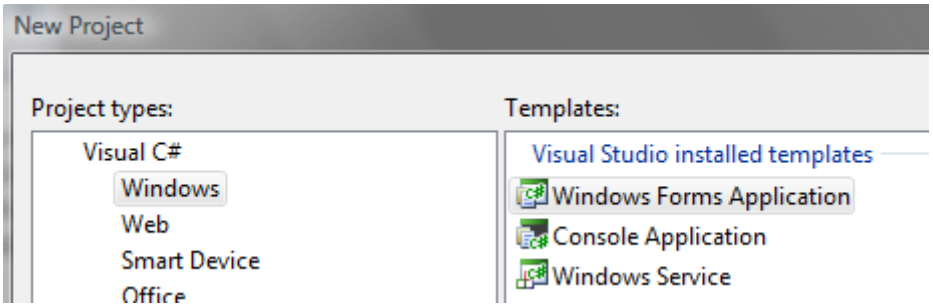
je využívaná u systémov používajúcich Windwos Froms, ASP.NET Web Forms, WPF, Java Swing a podobne. Tieto systémy si vedia užívateľský vstup ošetriť sami, napríklad vyvolanie udalosti na stlačenie tlačidla užívateľom, alebo zmena textu v textovom poli, či stlačenie klávesy na klávesnici. To znamená že view je schopné zachytiť užívateľský vstup.

Druhou variantov, ako môže užívateľ vstupovať do systému je pomocou Controller, kde sa spracováva užívateľský vstup. To znamená, že neexistujú žiadne komponenty na ošetrenie užívateľských vstupov do aplikácie.

Je dôležité uvedomiť si pozíciu modelu v MVC, pretože model znamená v skutočnosti doménový model, realizujúci vzťahy reálneho sveta. Dôležitou súčasťou nie sú len dáta ale aj vzťahy a skutočnosti v systéme. Keď to prenesieme do nami vytváraného systému UnIS, máme tým na mysli rôzne pravidlá, alebo obmedzenia užívateľov na vytváranie nového príjmového dokladu až vo chvíli ukončenia pridávania, alebo upravovania aktuálneho dokladu príjemky. V niektorých technológiách ako ASP.NET dochádza k definovaniu pravidiel do view, čo však potom nie je úplná architektúra MVC. Týmto krokom dochádza k duplikovaniu kódu pri podobných funkciách vkladania, alebo upravovania. Práve tomuto spôsobu sa vytváraná aplikácia UnIS vyhýba a na validáciu formulárov využíva vlastnú knižnicu, ktorá tieto služby poskytuje všetkým triedam projektu [2][3].

4.1.1 Vytvorenie klienta ako Windows Forms aplikáciu

Ako je už zrejmé z názvu tejto kapitoly, budeme sa zaoberať jednotlivými krokmi pri budovaní klienta DA pre desktopové aplikácie. Prvým krokom bolo vytvorenie projektu v Microsoft Visual Studiu. Tento projekt bol typu Windows, konkrétne Windows Forms Application ako môžete vidieť na obrázku.



Obrázok 16: Visual Studio - Windows Forms Application

Pre správne fungovanie klienta a umožnenie komunikácie so serverom prostredníctvom webových služieb, bolo potrebné pripojiť referencie na jednotlivé webové služby. Samotné pripojenie sme uskutočnili pomocou Add Service Reference, kde po zadaní adresy webovej služby, v našom prípade Services in Solution, bola pridaná referencia a mohli sme tak využívať poskytovanú funkcionalitu. Samotné vytvorenie poskytovania služieb bude popísané v kapitole 4.2.1Webové služby.

Samotným pridaním referencií ešte nie je zaistená komunikácia so serverom a poskytovanie služieb. V klientovi bolo potrebné umožniť nastavovanie pripájacieho reťazca „adresy“ k webovým službám. Tento reťazec na pripojenie má každý klient uchovaný vo svojich Properties a má možnosť ho meniť, aby pri nasadení mohol prispôsobiť nastavenie podľa vlastných potrieb. Nasledujúci kód predstavuje zmenu reťazca na pripojenie klienta k webovým službám:

```
      Properties.Settings.Default.URLtoWebServices = "URL reťazec";
      Properties.Settings.Default.Save();
```

Zdrojový kód 1: Nastavenie URL webových služieb

Aby sme mohli využívať metódy a funkcie pre tabuľku Firma v databáze, museli sme vytvoriť objekt typu NázovWebServiceSoapClient a nastaviť adresu URL z Properties klienta.

```
      rodic.URLWebServices = Properties.Settings.Default.
      URLtoWebServices.ToString();

...

      WebServicesFirmaSoapClient webRefFirm = new
      WebServicesFirmaSoapClient();
```

```
...

      webRefFirm.Endpoint.Address = new   System.ServiceModel.
      EndpointAddress(rodic.URLWebServices +    "/WebServicesFirma.asmx");

...

      UnISklientDA.ServiceReferenceFirm.Firm f = new
      UnISklientDA.ServiceReferenceFirm.Firm();
      f.Firm_Id = Id_Firm
      f.Firm_Name = textBox1.Text;

...

      UnISklientDA.ServiceReferenceFirm.Address a = new
      UnISklientDA.ServiceReferenceFirm.Address();
      a.Address_Id = Id_Address;
      a.Address_Street = textBoxStreet.Text;

...

      webRefFirm.UpdateFirmAndAddress(f, a);
```

Zdrojový kód 2: Použitie webových služieb

Zo zdrojového kódu reprezentujúceho úpravu firmy s vlastnou adresou je vidieť jednoduchý prístup k webovým metódam služieb cez bodkovú notáciu vytvoreného objektu, ako aj k jednotlivým objektom referencií na webové služby.

Čo sa týka užívateľského rozhrania, boli použité komponenty a operácie, ktoré poskytuje .NET Framework 3.0. Keďže bolo nutné užívateľské rozhranie aplikácie vtesnať do jedného okna, kde podľa požiadaviek zadávateľa má mať užívateľ „čašník“ možnosť spravovať viacero účtov naraz, bolo potrebné využiť hierarchické usporiadanie okien v aplikácií. Systém má umožňovať rýchlu navigáciu a správu databázy vo všetkých oblastiach. Je chránený prístup k viacnásobnému otváraniu totožných okien v aplikácií, no táto funkcia pre vytváranie reštauračných účtov, ktoré môžu byť v otvorenom stave aj viaceré, je vypnutá.

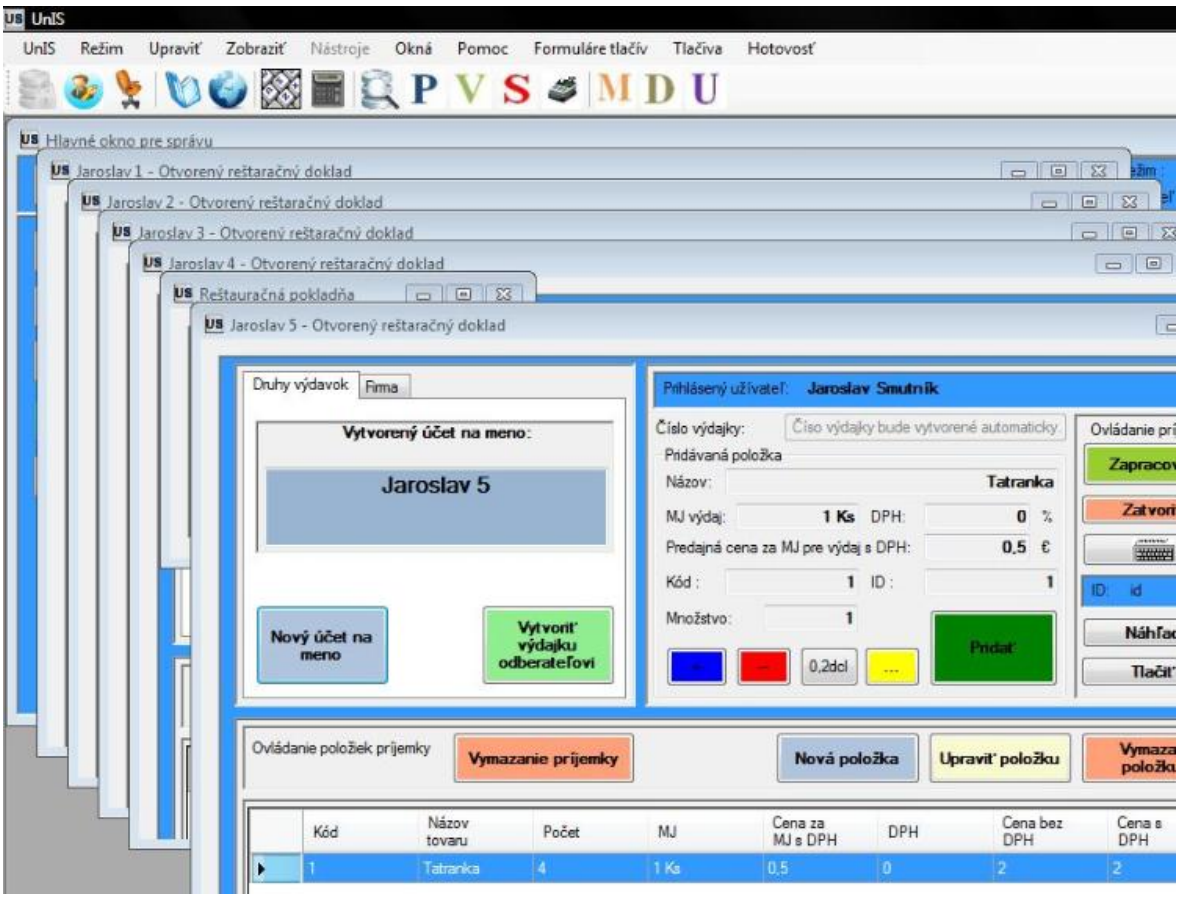
Nasledujúci obrázok umožňuje lepšie pochopenie hierarchie ukladania a otvárania okien. Pre vytváranie takejto aplikácie, ktorej okná nebudú roztrúsené po celej lište operačného systému, bolo potrebné vytvoriť jedno hlavné okno s navigačným menu a ostatné okná spúšťať v rámci tohto okna. Teda rodičom je hlavné okno programu, ktoré obsahuje svojich potomkov.

```
      //Vytvorenie hlavného (rodičovského) okna UnIS
      RUN.UnIS rodic = new RUN.UnIS();
      Rodic.Show();

...

      //Vytvorenie vnútorného okna Firma (potomok)
      Firm.Detail obFirm = new Firm.Detail(true);
```

```
        obFirm.rodic = rodic;
        obFirm.MdiParent = rodic;
        obFirm.Show();
```

Zdrojový kód 3: Hlavné okno a jeho potomkovia

Tento kód nám zaistí vytvorenie nasledujúcej hierarchie usporiadania otvorených okien v hlavnom okne.



Obrázok 17: UnIS klient DA - kaskádové zobrazenie okien

4.1.2 Tlačové zostavy

V tejto časti sa budeme venovať vytváraniu tlačových zostáv zo systému. Funkcia vytvárania tlačových zostáv je rozširujúcou funkcionalitou programu UnIS a ako sa ďalej zmienime, je poskytovaná len vo verzii klienta DA na Windows. Dôvodom je nekompatibilita s Mono Framework platformou.

Pre vytváranie tlačových zostáv sme použili PDFsharp and MigraDoc. Program umožňuje náhľad na vyrenderované zostavy priamo v spustenom programe a tiež vytláčanie zostáv aj bez nutnosti nainštalovaného prehliadača PDF súborov (Acrobat Reader).

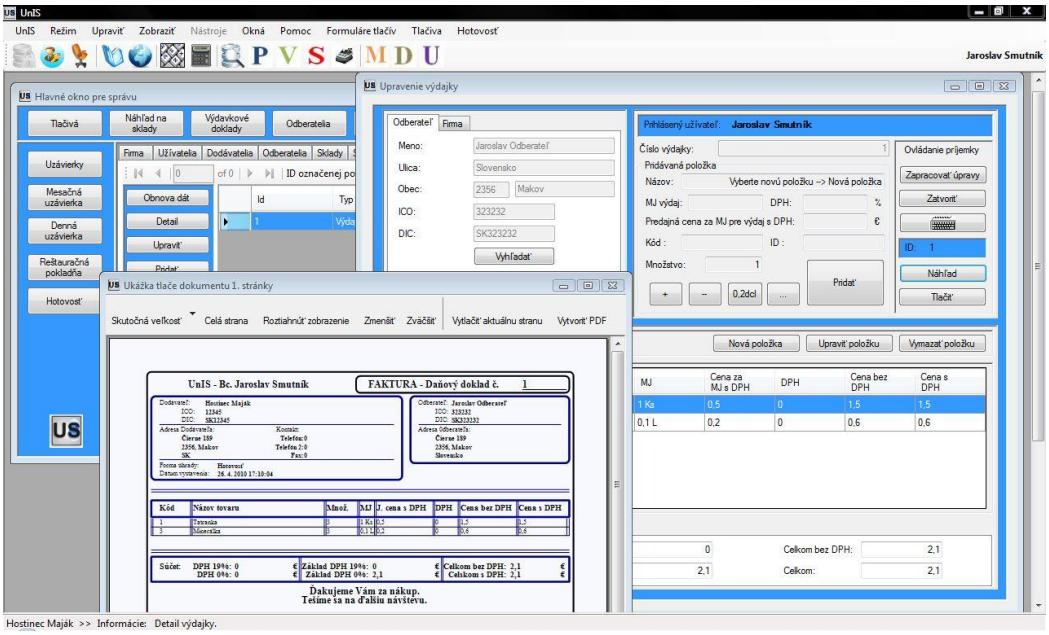
PDFsharp

Jedná sa o Open Source. NET knižnicu, ktorá je určená k ľahkému vytváraniu a spracovaniu PDF dokumentov z akéhokoľvek .NET jazyka. Používa sa k vytváraniu PDF dokumentov, zobrazovaných priamo v obrazovkách aplikácie, alebo vytváranie výstupov pre akékoľvek tlačiarne.

MigraDoc

Open Source .NET knižnica, ktorá umožňuje ľahko vytvárať dokumenty založené na objektovom prístupe, tabuľkách, štýloch, atď. Umožňuje vygenerovanie dokumentov do formátov PDF, XPS alebo RTF. Dostupná je na adrese: http://www.pdfsharp.net/Default.aspx.

Tieto knižnice sú Open Source, teda dostupné pre každého, bez poplatkov za licencie. Použitie budeme reprezentovať na vytváraní tlačových zostáv pre výdajky s odberateľom.



Obrázok 18: UnIS klient DA - Výdajka s odberateľom

Na obrázku predstavujúcom náhľad na vytvorenú výdajku s odberateľom, je vidieť vygenerovanie dokumentu v otvorenom okne ukážky tlače. Toto okno ponúka zobrazenú výstupnú zostavu vytlačiť priamo zo systému UnIS, alebo vygenerovať dokument vo formáte PFD, ktorý je schopná vytlačiť každá tlačiareň.

O vygenerovanie zobrazeného dokumentu (Daňový doklad) sa stará PDFsharp, za pomoci ktorého sme vytvorili šablónu, ktorú sme potom napĺňali dátami z formulára vytvorenej výdajky, alebo z databázy za použitia pripojenia na server.

```
/// <summary>
/// Vyrenderovanie výdajky za pomoci PDFsharp
/// </summary>
public void RenderVydajka(XGraphics gfx)
{
      NastavenieServices();
      XRect rect;
      XPen pen;
      double x = 50, y = 55;
      XFont fontH1 = new XFont("Times", 12, XFontStyle.Bold);

...

      double ls = font.GetHeight(gfx);

      // Vykreslenie štvorca okolo formulára dodávateľa
      rect = new XRect(x-5, y-15, 510, 770);
      pen = new XPen(XColors.Black, 2);
      gfx.DrawRoundedRectangle(pen, rect, new XSize(10, 10));

      // Vykreslenie nadpisu formulára
      gfx.DrawString("UnIS - Bc. Jaroslav Smutník",fontH1,
      XBrushes.Black, x + 45, y+3);

...

      // Vykreslenie údajov o dodávateľovi z centrálnej databázy
      // za pomoci webových služieb
      UnISklientDA.ServiceReferenceFirm.Firm f = new
      UnISklientDA.ServiceReferenceFirm.Firm();
      f.Firm_Id = bill.Firm_Id;

      //List s atribútmi firmy
      List<UnISklientDA.ServiceReferenceFirm.Firm> listFirm = new
      List<UnISklientDA.ServiceReferenceFirm.Firm>(webRefFirm.
      GetFirmWithAddress_BY_Firm_Id(f));

      //Vykreslenie názvu firmy z Listu
      gfx.DrawString("Dodávateľ:", font, XBrushes.Black, x + 10,y);
      gfx.DrawString(listFirm[0].Firm_Name.ToString(), fontBold,
      XBrushes.Black, x + 65, y);
      y += ls + 1;
```

Zdrojový kód 4: Vyrenderovanie zostavy pomocou PDFsharp

Pre lepšie pochopenie a vysvetlenie jednotlivých časti pri vytváraní dokumentov nám pomáhali veľmi dobre spracované stránky: http://www.pdfsharp.net/wiki/MainPage.ashx s množstvom názorných príkladov.

Vytvorenie PDF dokumntu bolo realizované za pomoci kombinácie MigraDoc a PDFsharp knižíc. Podstata je znázornená na následujúcom zdrojovom kóde:

```
/// <summary>
/// Vytvorenie tlačiva pre Účet s Odberateľom
/// </summary>
public void CreateBillFirstPage(PdfDocument doc, int pocetStran,
string Typ)
{
      NastavenieServices();

      PdfPage page = doc.AddPage();
      XGraphics gfx = XGraphics.FromPdfPage(page);

      gfx.MUH = PdfFontEncoding.Unicode;
      gfx.MFEH = PdfFontEmbedding.Default;

      XRect rect;

...//Pokračovanie kódu sa zhoduje s predchádzajúcou ukážkou
```

Zdrojový kód 5: Kombinácia PDFsharp a MigraDoc

Generovanie dokumentu sa líšilo len vo využití stránkovania vytváraného dokumentu PDF pomocou PdfDocument.AddPage(). Je to z dôvodu prípadného účtu, ktorý by mohol prekračovať svojimi záznamami jednu celú stranu, alebo viacero strán.

Bude možné zvoliť uloženie vygenerovaného tlačového výstupu do súboru PDF. Na uloženie vytvoreného dokumentu sme použili:

```
PdfDocument doc = new PdfDocument();
      doc.Info.Author = "Bc. Jaroslav Smutník";

...

      CreateBillFirstPage(doc, i ,"onePage");

      //Uloženie
      doc.Save(name);

      //Spustenie prehliadania uloženého súboru
      Process.Start(Environment.CurrentDirectory +
      System.IO.Path.DirectorySeparatorChar + "UnIS" +
      System.IO.Path.DirectorySeparatorChar + "Documents" +
      System.IO.Path.DirectorySeparatorChar + "Vydajky" +
      System.IO.Path.DirectorySeparatorChar + name);
```

Zdrojový kód 6: Uloženie a spustenie vytvoreného dokumentu PDF

4.1.3 Validácia formulárov

Na validovanie formulárov v klientovi DA sme vytvorili vlastnú knižnicu ClassLibraryClientUnIS, ktorá obsahuje metódy validovania. Bolo potrebné pridať referenciu na danú knižnicu ku klientovi DA systému UnIS. Po pridaní referencie bolo možné pracovať s objektmi knižnice, ktoré poskytujú metódy typu:

```
public Boolean ValidateNumber(string text)
{
      Regex regex = new Regex("^[0-9]*[,]?[0-9]*$");
      if (regex.IsMatch(text))
      {
            return true;
      }
      else
      {
            return false;
      }
}
```

Zdrojový kód 7: Validácia vstupu – čísla

Zobrazený zdrojový kód využíva regulárny výraz pre porovnanie vstupného reťazca text a vracia hodnotu typu Boolean.

4.2 Implementácia server

Serverová časť aplikácie bola vytvorená z dvoch projektov a to:

- **WebServiceUnIS** obsahujúci jednotlivé webové služby, umožňujúce komunikáciu medzi klientmi a serverom.

- **ClassLibraryServerUnIS**, ktorý sa stará o pripojenie k databáze, a tak umožňuje získanie prístupu k požadovaným dátam, ktoré sú prostredníctvom webových služieb poskytované klientom.

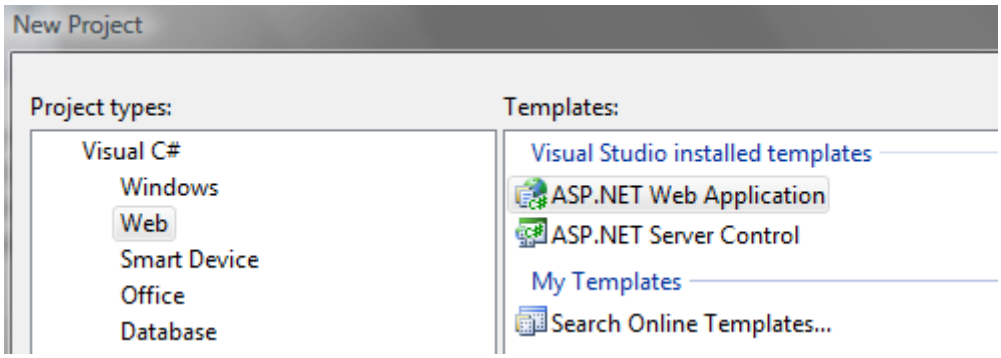
Projekty budú z pohľadu implementácie popísané v nasledujúcich kapitolách, spolu s použitou databázou pre systém UnIS.

4.2.1 Webové služby

Webovú službu môžeme označiť aj ako základný kameň budúcich internetových distribuovaných aplikácií. Ich účelom je umožňovať zdieľanie obchodnej logiky po sieti. Táto myšlienka bola využívaná už v minulosti, no narážala na niekoľko problémov s firewalmi a distribúciou v sieti. Dnešné využívanie webových služieb má veľký úspech a ten je založený na komunikácií pomocou protokolu http a na veľmi rozšírenom formáte XML, pomocou ktorého vymieňa dáta. Povolenie portov 80 (HTTP) na väčšine firewaloch umožňuje nerušenú komunikáciu aplikácií medzi sebou.

Teda webová služba je akousi aplikačnou logikou, ktorá je dostupná cez štandardné webové protokoly ako http alebo SMTP. Široké možnosti ponúka protokol Simple Object Access Protokol (SOAP), ktorý vznikol spojením HTTP a XML a umožňuje predávanie štruktúrovaných dát.

Vytvorený projekt v Microsoft Visual Studio je typu Web, konkrétne sa jedná o ASP.NET Web Application.



Obrázok 19: Visual Studio - Projekt Web Application

Samotná tvorba webových služieb nie je vôbec zložitá, stačí len odvodenie z triedy System.Web.Services.WebService, o ktoré sa postaralo samotné Visual Studio, keď sme do projektu pridali novú webovú službu WebService.asmx pomocou Add/New Item.

Nasledujúci kód popisuje zadefinovanie triedy pre prácu s webovými službami pre tabuľku firmy v databáze.

```
using System.Web.Services;

...

/// <summary>
/// Webová služba slúžiaca na správu firiem a získavanie dát o
/// firmách
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[ToolboxItem(false)]

public class WebServicesFirma : System.Web.Services.WebService
{
```

Zdrojový kód 8: Zadefinovanie triedy pre webové služby

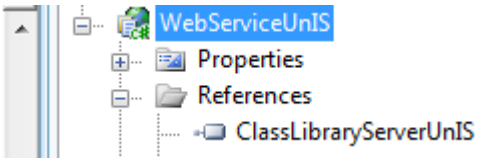
Aplikačnú logiku poskytovanú webovými službami tvoria verejné metódy prístupné cez webové rozhranie. Aby však boli prístupné ako webová služba, musia obsahovať atribút [WebMethod], ten zaistí sprístupnenie metódy ako jednej, konkrétnej webovej služby poskytovanej serverom. Tieto metódy môžu prijímať, odosielať a ďalej spracovávať požadovaný obsah, pričom sa automaticky starajú o prevod dát do a z XML.

```
/// <summary>
/// Zistenie existenie názvu firmy
/// </summary>
/// <returns>True/false či názov firmy existuje</returns>
[WebMethod]
public int Firm_Existence()
{
      Firm f = new Firm();
      return f.Firm_Existence();
}
```

Zdrojový kód 9: Ukážka vytvorenia webovej služby

V našom prípade, ako je zjavné zo zobrazeného zdrojového kódu webovej služby, sme využili vytvorenú knižnicu (ClassLibraryServerUnIS), ktorá slúži na komunikáciu s databázou. Týmto spôsobom webové služby vytvárajú objekty obsahujúce funkcionalitu serverovej časti a poskytujú ju webovým rozhraním klientom.

Pre využívanie vytvorenej knižnice, ktorá bude opísaná v nasledujúcej kapitole, bolo potrebné prideliť referenciu do projektu k webovým službám.

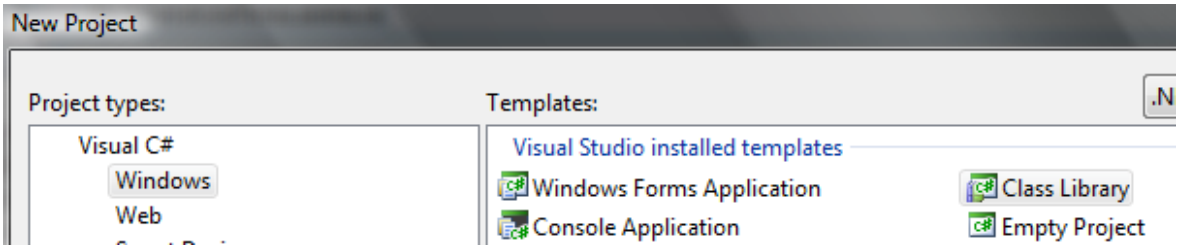


Obrázok 20: Visual Studio - referencia na knižnicu, pracujúcu s databázou

4.2.2 Vlastná knižnica servera

Pre prácu s dátami, uloženými v hlavnej centrálnej databáze systému UnIS sme vytvorili vlastnú knižnicu ClassLibraryServerUnIS. Jej úlohou je vytvoriť spojenie s databázou a vykonávať pomocou SQL príkazov jednotlivé parametrizované príkazy, ktorých výsledky sú ďalej poskytované webovej službe.

Vytvorený projekt knižnice patrí do typu Windows zrealizovali sme ho pomocou Microsoft Visual Studia nasledovne:



Obrázok 21: Visual Studio - Nový projekt Class Library

Ako vstupy do metód, poskytujúcich funkcionalitu jadra systému UnIS, sme použili celé objekty, s ktorých atribútmi sme ďalej pracovali.

```
namespace ClassLibraryServerUnIS
{
      public class Firm : Address
      {
      /**
      * Premenné pre firmu
      **/
      public virtual int Firm_Id { get; set; }
      public virtual string Firm_Name { get; set; }

...

      /**
      * SQL dotazy pre firmu
      **/
      //Update
      private const string UPDATE_FIRM = "UPDATE Firm SET
      Firm_Name=@Firm_Name, Firm_ICO=@Firm_ICO, Firm_DIC=@Firm_DIC,
      Firm_Tel=@Firm_Tel, Firm_Tel2=@Firm_Tel2, Firm_Fax=@Firm_Fax,
      Firm_Fax2=@Firm_Fax2 WHERE Firm_Id = @Firm_Id";

...

      public Boolean UpdateFirm(Firm f)
      {
            try
            {
                  using (SqlConnection myConnection = new
                  SqlConnection(ConnectionString))
                  {
                        SqlCommand myCommand = new
                        SqlCommand(UPDATE_FIRM, myConnection);

                        myCommand.Parameters.AddWithValue
                        ("@Firm_Id", f.Firm_Id);

                  ...

                        myConnection.Open();
                        myCommand.ExecuteNonQuery();
                  }
                  return true;
            }
            catch
            {
                  return false;
            }
      }
```

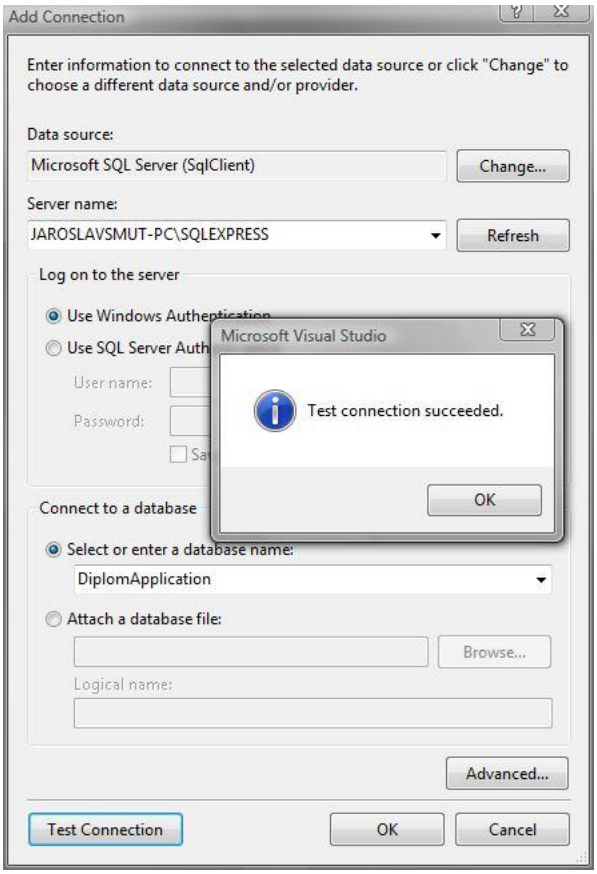
Zdrojový kód 10: Nadviazanie spojenia a úprava dát v databáze

Na príklade zdrojového kódu sme vytvorili spojenie s databázou SQLConnection, kde za parameter bol dosadený connection string, uložený v Settings knižnice. Na pridelenie SQL dotazu sme použili SQLCommand a na vloženie parametrov do dotazu myCommand.Parameters.AddWithValue() s atribútmi z objektu Firm f (f.Firm_Id).

4.2.3 Databáza

Základný stavebný kameň systému UnIS je databáza spravovaná serverom UnIS. Vytváranie hierarchie databázového systému UnIS bolo zrealizované za pomoci Microsoft SQL Management Studia, v ktorom sme nadefinovali jednotlivé tabuľky, priradili im atribúty a vytvorili prepojenia pomocou cudzích kľúčov v iných tabuľkách. Vytvorené väzby medzi tabuľkami nám poskytli kontrolu pri vymazávaní niektorých položiek obsahujúcich hodnotu primárneho kľúča, ako cudzieho kľúča.

Pre uľahčenie práce s vytvorenou databázou poskytuje Microsoft Visual Studio funkcionalitu podobnú samotnému SQL Management Studiu. Podstatné bolo vytvorenie samotnej databázy na SQL Serveri a potom vo Visual Studiu nastavenie pripojenia k databáze, ktorú sme používali a upravovali už prostredníctvom Visual Studia.



Obrázok 22: Visual Studio - Pridanie pripojenia na databázu

Nastavenie connection stringu

V serverovej časti aplikácie bol nastavený reťazec pre pripojenie k databázovému serveru a konkrétne k databáze reprezentujúcej UnIS. Connection string sa získava zo Settings v Properties implementácie knižnice servera nasledovne:

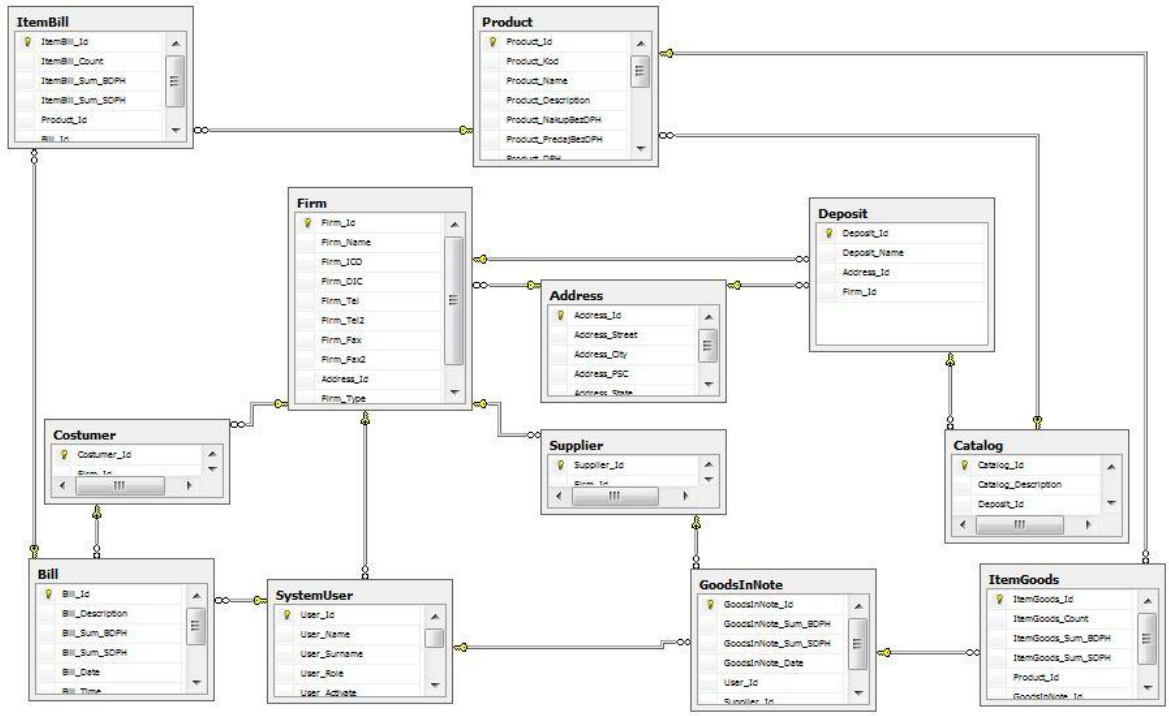
```
ClassLibraryServerUnIS.Properties.Settings.Default.
DiplomApplicationConnectionString;
```

Zdrojový kód 11: Získanie connection string na databázu

Podstatný krok, teda pripojenie servera k databáze je zaistený, nasleduje vytvorenie funkcionality serverovej časti, kde patria jednotlivé SQL parametrizované príkazy nad vytvoreným spojením k databázovému serveru boli popísané vyššie.

Pre vytvorenie hierarchie databázy je poskytnutý skript do Microsoft SQL servera, nachádza sa na priloženom CD v oddiely (Databáza). Tu je aj skript pre vytvorenie tabuliek, väzieb a naplnenie testovacími dátami v oddiely (Databáza\MSSQL\Skript pre naplnenie DB testovacími dátami).

Na obrázku je znázornená hierarchia vytvorenej databázy:



Obrázok 23: Hierarchia databázy

4.3 Implementácia klienta MA

Stále rastúce využívanie mobilných zariadení bolo jedným z dôvodov vytvorenia klienta MA s využitím .NET Compact Framework. Podporovanou platformou pre beh mobilnej aplikácie UnIS je Windows Mobile 5.0 Pocket PC SDK. Pocket PC charakterizuje hardwarovú špecifikáciu pre vreckové počítače, označované ako Personal digital assistant (PDA) s operačným systémom Microsoft Windows Mobile.

PDA

Býva zvyčajne ovládaný dotykovou obrazovkou, na ktorú sa používa špeciálne pero, nazývané stylus. Pôvodne bolo určené PDA zariadenie na usporiadanie času, úloh, alebo kontaktov užívateľa, teda ako náhrada zápisníka, či diára. Neskôr sa toto zariadenie stalo veľmi výkonným a spoľahlivým nástrojom, schopným zvládnuť aj náročnejšie operácie. Dnešná spoločnosť, ktorá si vyžaduje komfort a uľahčenie práce podporuje vývoj rôznych systémov pre tieto zariadenia. Systémová podpora zahŕňa Windows Mobile, Palm OS, Symbian OS, ale je možné nastaviť zariadenie aj na Linux. Základnou charakteristickou vlastnosťou je ľahkosť a prenositeľnosť, pretože váha sa pohybuje v rozmedzí 100 až 250 gramov.

Testy vytvorenej aplikácie boli prevádzané na emulátore USA Windows Mobile 5.0 Pocket PC R2 Emulator, ktorý poskytuje Microsoft Visual Studio 2008 Version 9.0.21022.8 RMT a na PDA zariadení HP iPAQ 114 s operačným systémom Windows Mobile® 6 Classic.

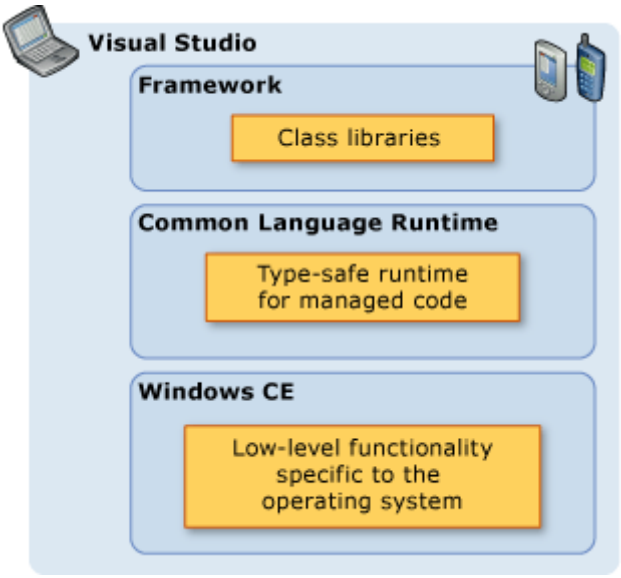
Ďalšie časti tejto kapitoly obsahujú podrobnejšie návody, ako vytvoriť mobilného klienta pokladnice s vlastnou databázou a využívaním webových služieb pre synchronizáciu vlastnej databázy s centrálnou databázou systému UnIS.

4.3.1 .NET Compact Framework

Microsoft. NET Compact Framework poskytuje rámec pre vytváranie klientských aplikácií, určených výkonným zariadeniam Pocket PC. .NET Compact Framework je v podstate podmnožinou spomínaného .NET Framework, prináša inteligentné riadenie vytváraného kódu. Sem patrí poskytovanie základných služieb pre správu pamäte, bezpečnosť ako aj funkcie operačného systému, čo umožní využívať tieto funkcie pomocou štandardného rozhrania API.

Architektúra .NET Compact Framework má primárne dve zložky a to Common Language Runtime a .NET Compact Framework Class Library.

.NET Compact Framework Class Library predstavuje skupinu rozhraní API určených pre tvorbu aplikácií.



Obrázok 24: Architektúr .NET Compact Framework

Medzi niektoré kľúčové Class Library sú:

- **Windows Form** – pre vývoj klientskych Windows aplikácií,
- **Základné triedy** – umožňujú pokročilé funkcie ako práca s vláknami, sieťovými zdrojmi a ďalšie,
- **GDI** – základnú podporu pre GDI, pri vytváraní grafiky,
- **Data a XML** – umožňujú jednoduchú manipuláciu s dátami a XML,
- **Web Service** – predstavujú podporu webových služieb klientov.

Aj keď je .NET Compact Framework podmnožinou .NET Framework, nie všetky funkcie podporuje .NET Compact Framework.

Pri tvorbe aplikácií na tejto platforme by ste mali brať ohľad na použitie grafického rozhrania. Kvôli veľkosti zariadenia je potrebné udržiavať prehľadnosť a jednoduchosť ovládania a nevyužívať mnoho prvkov v okne, aby bolo umožnené ovládanie dotykom.

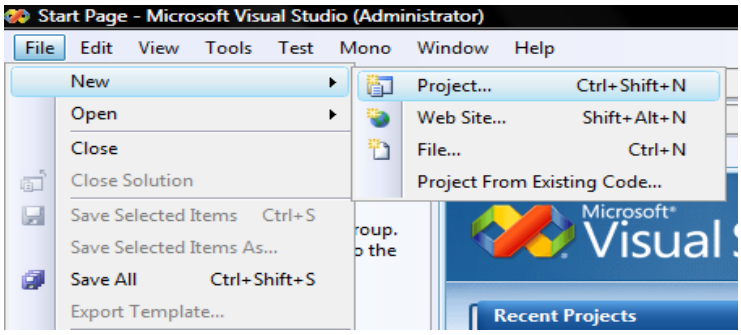
Čo sa týka výkonnosti týchto zariadení, je potrebné využívať jednoduchú funkčnosť, preddefinované funkcie a Multi Threading. Teda vytváranie kódu, ktorý nebude systém spomaľovať.

Pri použití databázy a dát uložených v zariadení, nezabudnite na obmedzenie pripojenia SQLCE, len na jedno pripojenie. Pri nasadení aplikácie sa vyhnite ActiveSync [25].

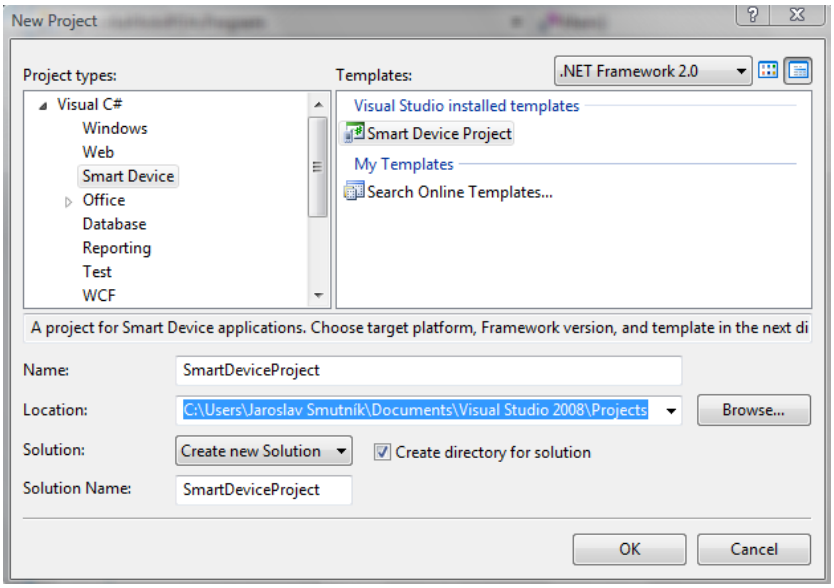
4.3.2 Smart Device projekt

V tejto kapitole ukážeme postupy a potrebné nastavenia pri vytváraní klienta MA pre UnIS, pričom budeme vychádzať z použitia Microsoft Visual Studio 2008 na realizáciu projektu. Programovací jazyk, v ktorom sú dostupné ukážky je C#.

Ako prvý krok bolo potrebné vytvoriť nový projekt typu Smart Device Project. Microsoft Visual Studio sa postaralo o vytvorenie potrebných nastavení a my sme tak mohli začať vytvárať jednotlivé okná s priraďovaním funkcií.



Obrázok 25: Navigácia vytvorenia smart device projektu



Obrázok 26: Vytvorenie smart device projektu

Klient MA pre UnIS si vyžadoval nastavenie rozhrania takým spôsobom, aby podľa aktuálneho stavu niektorých parametrov boli zobrazované tlačidlá, alebo neboli zobrazované tlačidlá, predstavujúce konkrétnu funkcionalitu. Tento problém sme vyriešili vytvorení špeciálnej triedy, cez ktorú

sme otvárali požadované okná a tá sa starala o správne zobrazovanie/skrývanie tlačidiel, podľa stavu atribútov prihlásenia, alebo stavu účtu.

Obrázok 27: UnIS PDA – Funkcionality neprihláseného a prihláseného užívateľa v UnIS pokladnici

Na zobrazenom obrázku hlavného okna pokladnice môžete vidieť poskytované funkcie na prácu s databázou tovaru, účtami a synchronizáciou v okne s prihláseným užívateľom. Pre zobrazenie alebo, skrytie tlačidiel a položiek menu sme použili:

```
      public Pokladna pokladna = null;

...

      public void ActivatePokladna(Boolean test)
      {
            //Ak existuje vytvorený objekt pokladna
            if (pokladna != null)
            {
                  //ButDatabaza je tlačidlo, definované v triede pokladna
                  pokladna.ButDatabaza.Visible = test;
                  //menuItemPrihlasit je položka menu v triede pokladna
                  pokladna.menuItemPrihlasit.Enabled = !test;
            }
      }
```

Zdrojový kód 12: Nastavenie práv v mobilnom zariadení

Na to, aby mohol klient využívať webové služby servera, museli sme vytvoriť webovú referenciu na požadované služby, ktoré obsahujú potrebné funkcie pre klienta tohto typu.

Pre lepšiu predstavu vytvorenia a realizácie spojenia klienta s centrálnou databázou servera za pomoci webových služieb, budeme ukazovať, ako sme umožnili aplikácií nastaviť pripojenie a získať dáta zo serveru. Túto funkcionalitu budeme popisovať na obrázku: Obrázok 28: UnIS PDA – Kontrola pripojenia do UnIS pokladnice.

Pre test pripojenia bolo nutné vytvoriť objekt webovej služby na firmu, ktorá obsahuje webovú metódu testu pripojenia a nastaviť URL požadovanej webovej služby pomocou premennej urlRetazec, v ktorom je reťazec na pripojenie, získaný z databázy klienta:

```
      WebServicesFirma.WebServicesFirma webRefFirma = new
      WebServicesFirma.WebServicesFirma();

...

      webRefFirma.Url = urlRetazec + "/WebServicesFirma.asmx";
      webRefFirma.TestWebServices();
```

Zdrojový kód 13: Nastavenie webových služieb v mobilnom zariadení

Obrázok 28: UnIS PDA – Kontrola pripojenia do UnIS pokladnice

Pokliaľ je testovanie neúspešné, užívatelia môžu reťazec v databáze mobilného klienta zmeniť. Umiestnenie reťazca je v databáze mobilného zariadenia, teda tu je uchovávaný pre daľšie pripojenie.

Ak je pripojenie k webovým službám nadviazané a užívatelia sú synchronizovaní s centrálnou databázou, môžeme prejsť k prihláseniu do systému a k práci s účtami.

Na zobrazovanie dát z databázy bola prevažne využívaná komponenta DataGrid. Jedná sa o Windows Forms DataGrid, ktorý poskytuje užívateľské rozhranie k ADO.NET DataSetom a umožňuje zobrazovanie ADO.NET dát v prehľadnej tabuľke, alebo upravovanie dátových zdrojov. Ak je DataGrid viazaný k zdroju dát s jednoduchou tabuľkou, neobsahujúcou žiadne vzťahy, dáta z tejto tabuľky sa zobrazia prehľadne v DataGrid komponente aj s názvami stĺpcov z databázovej tabuľky. DataGrid komponenty môžu byť použité na zobrazenie jednej tabuľky, alebo na zobrazenie hierarchických vzťahov medzi tabuľkami.

Príklad použitia a nastavenia DataGrid komponenty si ukážeme na okne Sklad tovaru, ktorý obsahuje DatGrid naplnený údajmi z tabuľky Product, uloženej v mobilnom zariadení.

Obrázok 29: UnIS PDA – Sklad tovaru v UnIS pokladnici

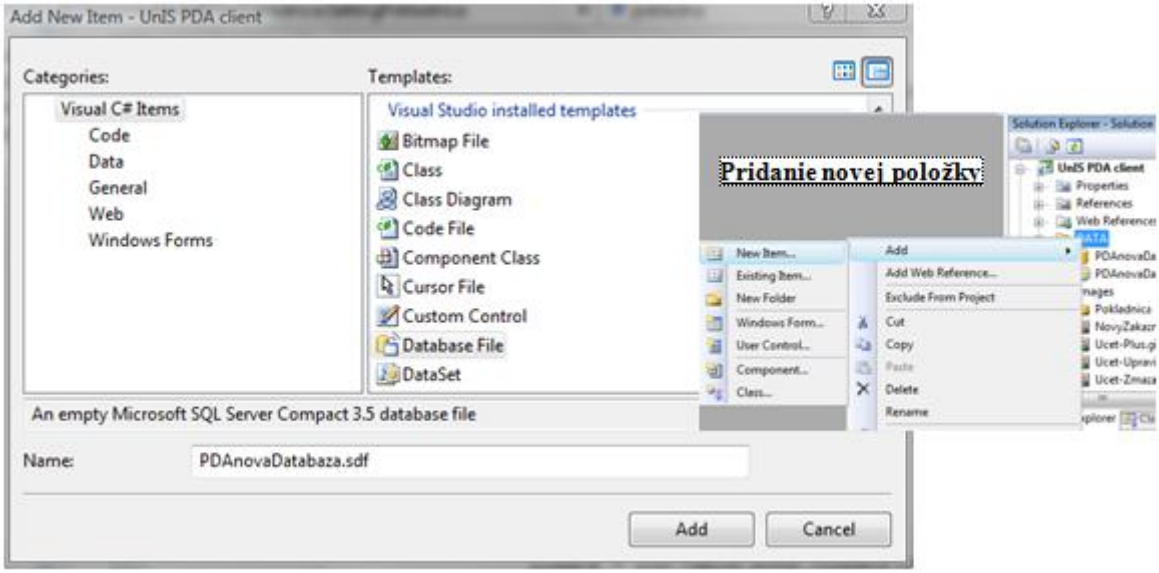
Na obrázku môžete vidieť tabuľku, v ktorej sú naplnené dáta a funkcie vyhľadávania, alebo prideľovania zvoleného množstva na existujúci účet.

Vytváranie formulárov a užívateľského rozhrania bolo veľmi intuitívne a dobre podporované Microsoft Visual Studio, ponúka množstvo vytvorených komponent, pripravených na použite v aplikáciách.

4.3.3 Realizácia databázy pre mobilné zariadenia

Jadrom UnIS klienta MA pre mobilné zariadenia je vlastná databáza. Databázový server určený nášmu mobilnému klientovi s platformou Microsoft Windows Mobile 6, je takisto vyprodukovaný firmou Microsoft a jeho názov je Microsoft SQL Server Compact (SQL CE). Jedná sa o kompaktnú relačnú databázu pre mobilné zariadenia a stolové počítače. Najnovšia verzia je SQL Server Compact 3.5 SP2 podporujúca .NET Framework 3.0 ako aj Windows Mobile 2003, 5.0, 6.0, 6.5.

SQL CE zdieľa spoločné API s ďalšími edíciami Microsoft SQL Server, zahŕňa ADO.NET na prístup k dátam používajúc ADO.NET API. Databázy bývajú uložené v jednom súbore s príponou .sdf, ktorý môže nadobudnúť veľkosť až 4GB. SDF súbor môže byť šifrovaný 128-bitovým kódovaním pre ochranu dát. Tento SDF súbor je možné jednoducho skopírovať do systému, v ktorom bude použitý alebo použiť službu Microsoft ClickOnce. Aplikácie používajúce SQL CE databázu nemusia špecifikovať celú cestu k súboru .sdf v connection stringu ADO.NET a nastavenie hesla pre databázový súbor je nepovinné [21].

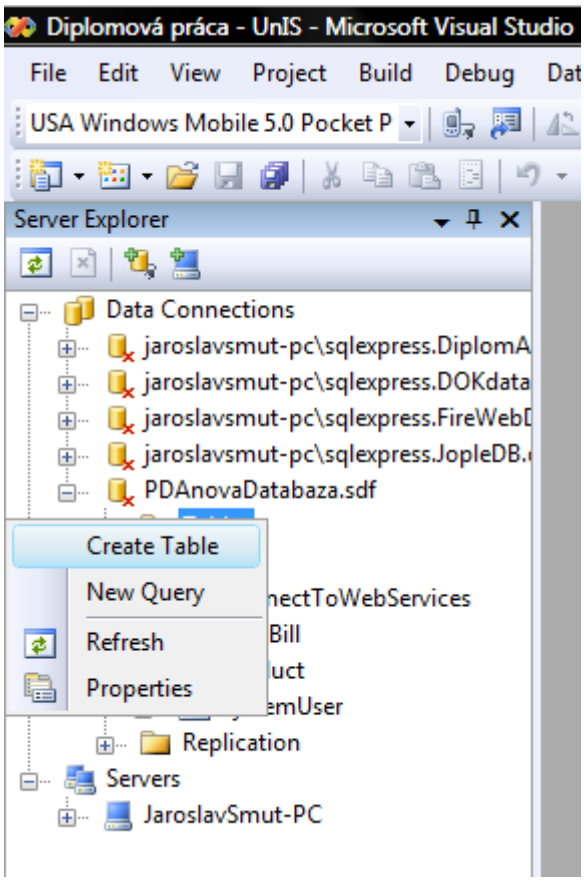


Obrázok 30: Visual Studio - Pridanie databázy

Na obrázku je zobrazený postup vytvorenia novej databázy do projektu mobilného zariadenia. Názov vytvorenej databázy je PDAnovaDatabaza.sdf a obsahuje hierarchiu tabuliek potrebných k funkcionalite klienta aj bez priameho kontaktu so serverom. To znamená nezávislé používanie mobilného klienta s vlastnou databázou, ktorá má možnosť synchronizácie dát s centrálnou databázou na serveri, ku ktorému sa pripojí klient pomocou webových služieb poskytovaných serverom UnIS.

Vytvorenie jednotlivých tabuliek potrebných k chodu databázy a nastavenie hierarchie databázového súboru je nasledovné:

- Dvojklik na názov vytvorenej databázy (PDAnovaDatabaza.sdf) v prehliadači serverov Visual Studia otvorí databázu a pravým tlačidlom myši je možnosť vytvorenia novej tabuľky (Create Table).

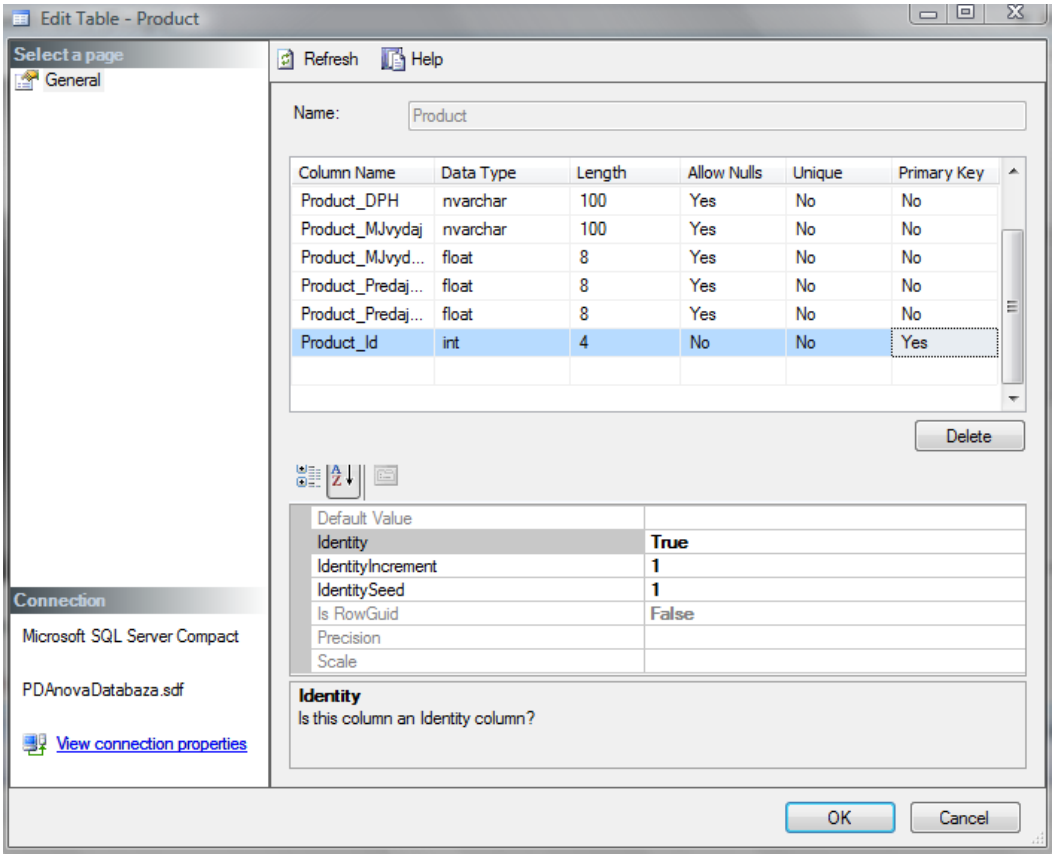


Obrázok 31: Visual Studio - Vytvorenie tabuľky

- Vytváranie jednotlivých tabuliek je obdobou zobrazeného príkladu na nasledujúcom obrázku, ktorý ukazuje editáciu tabuľky produkt s nadefinovanými atribútmi. Užívateľ si navolí jednotlivé atribúty, v našom prípade sa jedná o atribúty použité pre správny chod pokladnice a nastaví im hodnoty dátového typu (Data Type) a ostatné vlastnosti podľa potreby. Možnosť zvoliť primárny kľúč tabuľky umožní jednoznačný prístup k dátam. Pre uľahčenie správy indexov primárneho kľúča je možnosť výberu automatického pridávania s nastavením hodnoty skoku a jedinečnosti.

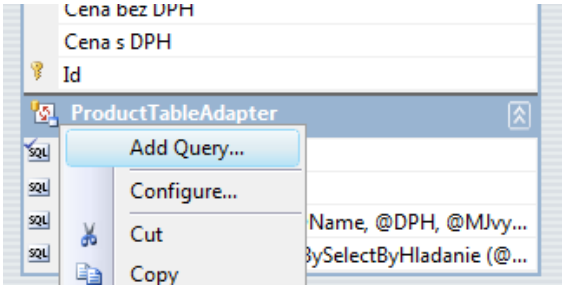
- Vytvorené tabuľky pomocou návrhára umožňujú pridávanie jednotlivých funkcií, ktoré budú potrebné na získavanie, upravovanie, vkladanie a zmazávanie dát. Postup pridávania SQL funkcií je zobrazený na obrázku: Obrázok 33: Visual Studio - Pridanie SQL funkcií.

Takýmto spôsobom sú nadefinované požadované operácie v systéme klienta pre mobilné zariadenia a využívané v kóde. Prístup k ním je prostredníctvom vytvoreného tabuľkového adaptéra (názovTableAdapter) k jednotlivý tabuľkám. Vytváranie funkcií je uľahčené editorom, ktorý automatizuje vytváranie sql parametrizovaných príkazov.



Obrázok 32: Visual Studio – Úprava existujúcej tabuľky

Vytvorené tabuľky sú uložené do súboru PDAnovaDatabaza.sdf a poskytujú funkcionalitu, teda jadro celého systému klientskej aplikácie.



Obrázok 33: Visual Studio - Pridanie SQL funkcií

Využitie vytvorených adaptérov v kóde pre načítavanie dát do spomínaných DataGrid tabuliek je nasledovné:

```
      /// <summary>
      /// Načítanie okna so zoznamom tovaru z databázy klienta
      /// </summary>
      private void DataTovar_Load(object sender, EventArgs e)
      {
            //Zobrazenie všetkého tovaru v tabuľke Product
            this.productTableAdapter.Fill(this.pDAnovaDatabazaDataSet.
            Product);
      }
```

Zdrojový kód 14: Naplnenie DataSet tabuľky pomocou TableAdapter v mobilnom zariadení

Z nasledujúceho kódu je zrejmé využitie vytvoreného adaptéru (productTableAdapter) pre konkrétnu tabuľku Product, pomocou ktorého sme vyvolali funkciu Fill pre naplnenie adaptéru dátami z tabuľky s parametrom objektu Product uloženého v DataSete.

Podobným spôsobom sú v tabuľke zobrazované dáta po vykonaní vyhľadávania za pomoci parametra. Vkladanie dát je umožnené pomocou tabuľkového adaptéra do databázy mobilného klienta nasledujúcim spôsobom, ktorý reprezentuje získanie názvu produktu podľa označeného riadku v tabuľke DataGrid a vloženie ďalšieho záznamu položky účtu s údajmi o produkte :

```
      //Získanie označeného riadka v DataGrid
      int riadokUprav = dataGridTovar.CurrentRowIndex;
      //Získanie počtu pridávaého tovaru z texBoxu
      double pocet = Convert.ToDouble(textBoxPocet.Text);
      //Získanie názvu označeného produktu z DataGrid tabuľky,
      //umiestnenom na pozícií riadku – riadokUprav a pozícií stĺpca – 1
      string productName = dataGridTovar[riadokUprav, 1].ToString();
      //Získanie Id označeného produktu z DataGrid
      int productId = Convert.ToInt32(dataGridTovar[riadokUprav, 7]);

...

      //Výpočet výslednej sumy na vytváraný účet
      sumaBDPH = (pocet * cenaBDPH);
      sumaSDPH = (pocet * cenaSDPH);

...

      //Vloženie záznamu položky účtu do databázy
      detailUctov.itemBillTableAdapter.InsertQuery(productName, pocet,
      sumaSDPH, sumaBDPH, productId, billId);
```

Zdrojový kód 15: Použitie DataGrid údajov v mobilnom zariadení

Už by malo byť zrejmé ako využívať ďalšie SQL funkcie, vytvorené v návrhárovi Microsoft Visual Studia. Ďalej ukážeme metódu na prepočítanie celkovej ceny účtu, pri pridaní položky a získanie potrebných údajov z DataGrid tabuľky. Ešte sme nepoznamenali, že pozície jednotlivých buniek riadkov a stĺpcov sú uvádzané od nuly.

Teraz spomínaný príklad funkcie:

```
      //Prechádzanie tabuľky položiek účtov a získanie údajov o ich
      //cenách
      for (int i = 0; i < detailUctov.dataGridPolozkaUcet. VisibleRowCount;
i++)
      {
            //Získanie celkovej ceny položiek účtu bez DPH
            CelSumaBDPH += Convert.ToDouble(detailUctov.
            dataGridPolozkaUcet[i, 3].ToString());
            //Získanie celkovej ceny polžiek účtu s DPH
            CelSumaSDPH += Convert.ToDouble(detailUctov.
            dataGridPolozkaUcet[i, 2].ToString());
      }

      //Využitie funkcie na úpravu celkovej ceny účtu, podľa jeho Id
      this.billTableAdapter1.UpdateSumBDPHAndSumSDPHbyBillID(Math.Round(
      CelSumaBDPH, 2), Math.Round(CelSumaSDPH, 2), Convert.ToInt32
      (detailUctov.idUctu));
```

Zdrojový kód 16: Využitie pozície buniek v DataGrid

Vytvorená databáza má všetky údaje o svojich účtoch zaznamenané v zozname účtov, ktoré môžu byť v troch stavoch. Prvým je stav „Nezaplateny“ ktorý reprezentuje otvorený účet a umožňuje meniť položky, alebo úplne zmazanie účtu. Druhým stavom je „Zaplateny“, ten umožňuje užívateľovi účty len prezerať a zakazuje ich meniť alebo mazať, do zmeny stavu na „Synchronizovany“, kde je možné účty zmazávať z databázy mobilného zariadenia, pretože sú už uložené do centrálnej databázy systému UnIS.

Obrázok 34: UnIS PDA – Zoznam účtov a porovnanie práv v závislosti na stave

5 Prenos WinForms aplikácií

Úsilie potrebné k prenosu WinForms aplikácií na Mono sa môže veľmi líšiť. Hoci mnoho aplikácií pobeží na Mono bez zmeny, existujú však aplikácie, ktoré si vyžadujú pre správny beh na Mono upravenie zdrojového kódu. V tejto kapitole popíšeme návod, ako preniesť aplikáciu vyvíjanú na platforme .NET Framework v Microsoft Visual Studiu. Cieľom je prenos na systém Linux s využitím Mono Framework.

5.1 Mono Migration Analyzer

Pre jednoduchý prenos aplikácií a zistenie stavu pripravenosti prenášanej aplikácie je poskytovaný program Mono Migration Analyzer (MoMA). Pomáha odhaliť problémy pri prenose aplikácie z .NET Framework na Mono Framework a tak určiť oblasti, ktoré ešte nie sú podporované v rámci projektu Mono. Tento nástroj nepokrýva úplne všetky možné chyby, ktoré môžu nastať pri prenose, preto treba vždy aplikáciu ešte otestovať na Mono Framework.

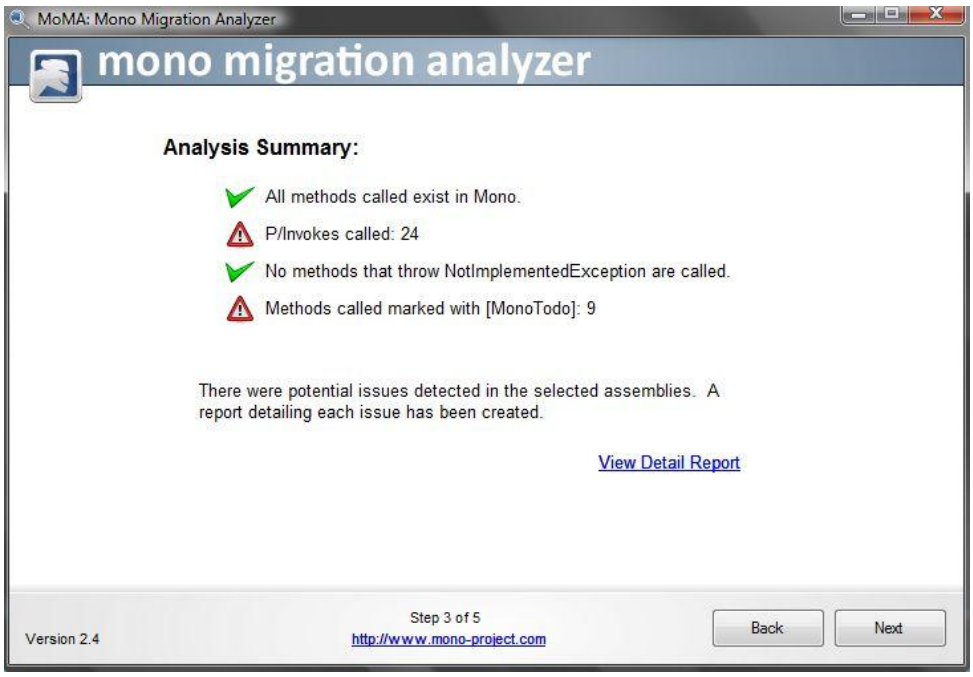
Problémy na ktoré môžete naraziť počas prenosu aplikácií medzi Windows a Linux (alebo OSX, atď) sú rozdelené na štyri typy:

- **Missing Methods** – chýbajúce metódy patria medzi najzávažnejší problém pri prenose,
    - **Riešenie** – jedinou možnosťou je odstránenie tejto metódy z vašej aplikácie a snaha nahradiť ju inou, ktorú podporuje Mono Framework.
- **MonoTodo** - metódy označené MonoTodo môžu, alebo nemusí spôsobiť problémy pre vašu aplikáciu. Môže to znamenať len čiastočnú podporu metód, alebo nemusia označené metódy vykonávať žiadnu funkciu,
    - **Riešenie –** tieto výstrahy môžete pri prvotnom prenose ignorovať a znamenajú len chýbajúcu funkcionalitu vašej aplikácie. Chýbajúce funkcie môžu byť opravené tým, že bude metóda dokončená v ďalšej verzii Mono .
- **NotImplementedException** – je zložité určiť, či nastane problém pri spustení aplikácie pod Mono Framework. Vo väčšine prípadov nie sú tieto metódy vôbec implementované a jednoducho vyhodia výnimku pri zavolaní,
    - **Riešenie** – nedá sa presne určiť, či je príčinou problémov MonoTodo. Preto treba aplikáciu otestovať aj iným spôsobom – spustením
- **P/Invokes** – sa používajú na volanie funkcií, ktoré sú napísané v neudržovaných jazykoch, často krát ich poskytuje sama platforma (user32.dll, Shell32.dll). Mono môže odchytiť tieto

volania, ak je knižnica podporovaná platformou, na ktorej beží program, avšak aplikácia tak prestáva byť multiplatformová, teda záleží na platforme, či tieto knižnice obsahuje,

- **Riešenie –** ak voláte niečo, čo poskytuje svoju platformu (zvyčajne Win32 API), budete musieť nájsť spôsob, ako dosiahnuť rovnaké funkcie pre všetky cieľové platformy. To by mohlo znamenať nahradenie takýchto metód, alebo zistenie o akú platformu sa jedná (Win32 / * nix / OSX / atď.) a podľa toho zavolať požadovanú funkciu na dane platforme.

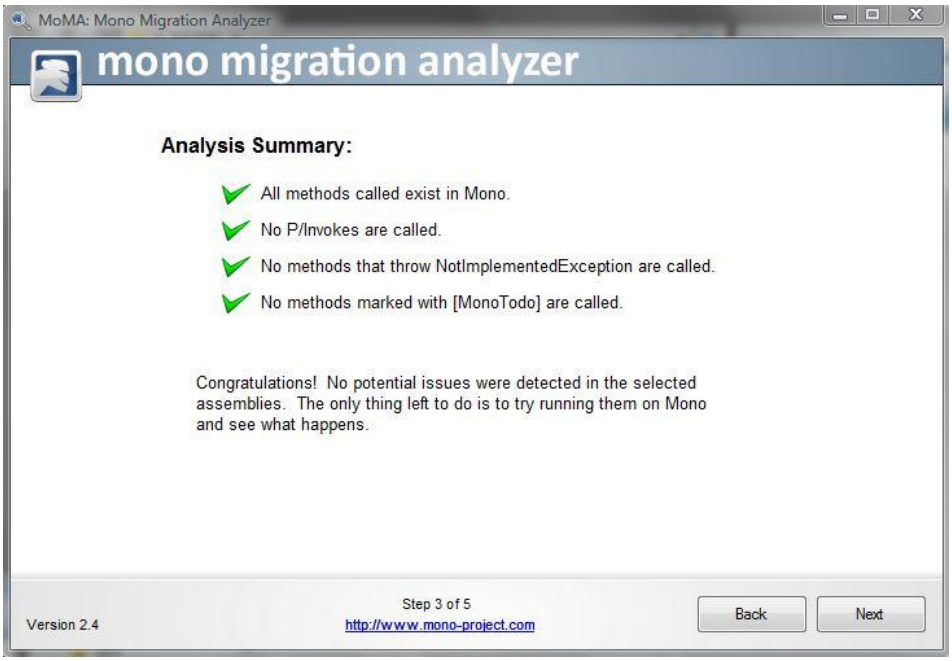
Čo sa týka našej aplikácie UnIS, výsledok po prevedení v programe MoMA bol nasledovný:



Obrázok 35: MoMA pre UnIS s PDF reportami

Vytvorená pôvodná aplikácia systému UnIS v .NET Framework bola rozšírená o tlačové výstupy vo formáte PDF. Z reportu, zobrazenom na obrázku v prílohe C, môžete vidieť, aké chyby vadia pri prenose aplikácie na Mono Framework.

Ako náprava boli preto vo verzii pre Mono Framework odstránené funkcie pre vytváranie a prehliadanie tlačových výstupov vo formáte PDF. Dôvodom bolo nedostupnosť v Mono Framework a chybovosť aplikácie pri spustení. Po odstránení tlačových výstupov, teda knižníc, ktoré robili problémy pri prenose (PDFsharp a MigraDoc) sme docielili úplnú kompatibilitu s Mono Framework, ktorej výsledok je znázornený na zobrazenom obrázku: Obrázok 36: MoMA pre UnIS bez PDF reportov [14][16].



Obrázok 36: MoMA pre UnIS bez PDF reportov

6 Nasadenie systému

Systém je pôvodne vyvíjaný na platforme .NET Framework v operačnom systéme Windows, no je očakávané nasadenie na platformu Mono Framework do operačného systému Linux. Komplikáciám, ktoré mohli nastať pri prenose aplikácie na inú platformu, sme sa snažili vyhýbať už pri vývoji. Kroky ako vytvárať aplikáciu, ktorá bude prenositeľná na Mono Framework a tým pádom na všetky podporované platformy tohto Frameworku, sme opísali už v rannej fáze tejto diplomovej práce. Teda v teoretickej časti, konkrétne v kapitole Prenositeľnosť aplikácií.

Čo sa týka jednotlivého nasadenia aplikácie, boli vybraté dva operačne systémy, ktoré využíva väčšina užívateľov na stolových, prenosných, alebo iných druhoch počítačov. Postupy nasadenia nášho UnIS pre obchod so skladom je popísaný v nasledujúcich kapitolách. Zvolený výber operačných systémov pre ukážku nasadenia mal dva dôvody. Prvým je nasadenie na najpoužívanejší operačný systém Windows z dôvodu vytvárania a testovania systému UnIS v tomto operačnom systéme. Druhým dôvodom je nutnosť kompatibility nášho systému s Mono Framework, ktorý je primárne určený pre použitie na Linux.

Prvé spustenie aplikácie nad testovacími dátami si vyžaduje zadanie užívateľského loginu: admin, a hesla: pass. Meno a heslo je rovnaké pre testovacie dáta v oboch databázach systému UnIS.

6.1 Nasadenie na Windows

Spoločnosť Microsoft je tvorca rady grafických operačných systémov nazývaných Microsoft Windows. Tieto systému sú zväčša určené pre osobné počítače a procesory Intel, nejedná sa však o pravidlo používania týchto systémov len na túto konfiguráciu. Podporované procesory sú 16bitové, 32bitové a 64bitové.

Výnimkou nie sú ani PDA zariadenia, ktoré sú naším systémom podporované a jedná sa o klienta MA pre mobilné zariadenia. Z minulosti sa vám môže zdať známy názov Windows Pocket PC Edition, neskôr však došlo k premenovaniu systémov pre tieto zariadenia a ich oficiálny názov v súčasnosti je Windows Mobile.

6.1.1 Beh aplikácie pod .NET Framework

.NET Framework

Prvým krokom je nainštalovanie .NET Framework na váš systém, ak ho už neobsahuje. Potrebná verzia je .NET Framework 3.5, ktorý je možné stiahnuť na stránkach spoločnosti Microsoft, v sekcii sťahovania softwaru: http://www.microsoft.com/downloads/en/default.aspx.

Databáza

Tvorí základný kameň celého systému. Jedná sa o centrálnu databázu servera aplikácie UnIS. Spomínali sme vytvorenie ďalšej databázy mobilného klienta, ktorej postup nasadenia bude obsiahnutý v nasledujúcej kapitole: 6.2 Nasadenie na Windows Mobile. Aby sme sa vrátili k centrálnej databáze, tak bol použitý Microsoft SQL Server 2005 a software je možné získať na spomínanej adrese spoločnosti Microsoft v úseku sťahovania softwaru. Pre uľahčenie práce a vytvorenie potrebnej databázy s hierarchiou systému UnIS je možné stiahnutie nástroja pre úpravu a správu Microsoft SQL Servera, ktorý sa nazýva Microsoft Server Management Studio Express. Za jeho pomoci sa nahrajú skripty pre vytvorenie databázy a naplnenie testovacími dátami. Tieto skripty sú obsiahnuté v prílohe tejto diplomovej práce na priloženom CD v oddiely (Databáza\MSSQL\Skript pre vytvorenie DB), alebo je tu dostupný celý databázový súbor DiplomApplication.mdf. Prípadne ak je potreba nahrania testovacích dát do vytvorenej databázy, je poskytnutý skript na priloženom CD v oddiely (Databáza\MSSQL\Skript pre naplnenie DB testovacími dátami) s názvom UnIS DB.sdf. Login a heslo testovacích dát sú: „admin“, „pass“.

Druhým spôsobom nasadenia databázy je pomocou databázového servera MySQL, postup pri nasadení je popísaný v kapitole: 6.3.3 Databáza MySQL.

IIS server

Pre beh servera a poskytovanie webových služieb je potrebné nainštalovať IIS server, ktorý umožní poskytovanie dát z databázy prostredníctvom vytvorených webových služieb. Knižnice obsahujúce webové služby a služby poskytované serverom sú uložené na priloženom CD v oddiely (Systém UnIS\UnIS server\.NET Framework - Windows). Je potrebné v aplikácií servera správne nastaviť connection string na databázu.

Spustenie klienta DA

Po nastavení servera a správneho poskytovania webových služieb môžete spustiť samotného klienta aplikácie UnIS. Nachádza sa na priloženom CD v oddiely (Systém UnIS\UnIS client\UnIS klient pre desktopové aplikácie\.NET Framework - Windows\RUN UnIS), kde je uložený pod názvom UnIS (Windows).exe. Ak budete chcieť spustiť aplikácie z vlastného počítača, je potrebné nakopírovať celý adresár RUN UnIS a z neho aplikáciu spúšťať, prípadne si vytvoriť odkaz na plochu z tohto adresára. Pri prvom spustení bude prevedená kontrola pripojenia klienta k webovým službám a kontrola správneho nastavenia servera, ktorý ma komunikovať s vytvorenou databázou. Po týchto kontrolách systém spustí svoju inštaláciu, teda bude požadovať od užívateľa vytvorenie hlavičky firma a administrátora, ktorý bude UnIS pre konkrétnu firmu spravovať. Až po týchto krokoch bude užívateľ pripustený do aplikácie, cez login a heslo administrátora, ktoré zadal pri inštalácií systému.

Pokiaľ si vyberiete nastavenie databázu servera s testovacími dátami, potom nebudete musieť prechádzať kroky inštalácie popísané v prílohe: B Inštalácie systému UnIS.

6.1.2 Beh aplikácie pod Mono Framework

Pre beh aplikácie v Mono Framework pod operačným systémom Microsoft Windows je nevyhnutná inštalácia Mono na Windows opísaná nižšie.

Inštalácia a spustenie Mono na Windows

Ako prvé je potrebné nainštalovanie Mono. Stiahnuť najnovšiu verziu pre Windows je možné z oficiálnych stránok projektu: http://www.go-mono.com/mono-downloads/download.html.

Spustením prevzatých súborov sa vám vyvolá inštalácia, v ktorej priebehu budete musieť odsúhlasiť licenčné práva a podmienky na používanie.

Nakoniec budete odpovedať na niekoľko štandardných otázok ako:

- do ktorého adresára sa má nainštalovať Mono,
- ktoré komponenty sa majú nainštalovať,
- či zaradiť inštalovaný produkt do menu,
- ktorý port bude využívať XSP (webový server pre Mono).

Po štandardnej inštalácií sa vytvorí nové „Mono pre Windows“ program v rámci skupiny Štart menu s odkazmi na všetky bežné nástroje, ktoré budete potrebovať, aby ste mohli začať s Mono pracovať. Vytvorený a nakonfigurovaný príkazový riadok nájdete pod hlavnou skupinou Mono.

Ak chcete otestovať MCS kompilátor a spustiť Mono, zadajte tento príkaz do príkazového riadka pre Mono. Ten vytvorí jednoduchý C# súbor:

```
C: \> echo triedy X (static void main ()
(System.Console.Write ("OK ");))> x.cs
```

Teraz môžete skompilovať výsledný "x.cs" súbor, s Mono C # kompilátorom:

```
C:\> mcs x.cs
```

Pre otestovanie spustenia vytvoreného (x.exe) súboru prostredníctvom Mono a Microsoft runtime, zadajte príkaz:

```
C:\> mono x.exe

OK
```

```
C:\>
```

Pre odskúšanie pod Windows zadajte príkaz bez mono:

```
C:\> x.exe

OK

C:\>
```

Ak dostanete rovnaký výsledok, máte správne nainštalovaný Mono Framework pod Windows.

Súčasťou inštalácie Mono je Gtk#, ktorý vám umožní vytvárať Gtk# aplikácie pre Windows s runtime Mono, ktoré môžete neskôr umiestniť na Linux.

Kombinovaný inštalátor vytvorí v adresári nainštalovaného programu zložku Aplications, ktorej obsahom sú Gtk# aplikácie umožňujúce spustenie testu.

Sú to:

- **Prj2Make# GTK** – jedná sa o grafické rozhranie k prj2make knižnici, ktorú môžete využiť na vytvorenie Makefile z Visual Studio .NET,
- **SQL# GTK** – je grafický analyzér, ktorý podporuje niekoľko rôznych databáz.

Prípadne, ak chcete používať Gtk# pre Windows, bez Mono, môžete použiť Gtk# inštalátor pre .NET Framework.

Kombinovaný inštalátor ešte vytvorí adresár XSP s odkazmi na spustenie XSP a XSP2. Čo sú Mono ASP.NET a ASP.NET 2.0 webové servery [19].

IIS server

Rovnako ako s databázou to môžete vyriešiť aj so serverom poskytujúcim webové služby. To znamená nastavenie IIS servera rovnakým spôsobom, ako v kapitole: 6.1.1 Beh aplikácie pod .NET Framework. Tým zaistíme beh celej serverovej časti na operačnom systéme Microsoft Windows a pripojenie na databázu tiež na rovnakom systéme.

Databáza

Databázu je možné požiť rovnakú ako bola v kapitole: 6.1.1 Beh aplikácie pod .NET Framework. Tým zaistíme beh serverovej časti na operačnom systéme Windows a poskytovanie webových služieb klientovi, ktorý bude bežať pod Mono Framework na Windows operačnom systéme.

Druhou možnosťou je nastavenie databázového serveru MySQL, ktoré je popísané v kapitole: 6.3.3 Databáza MySQL.

Spustenie klienta DA

Pre spustenie klienta DA v prostredím Mono Frameworku je potrebné mať nastavené predošlé kroky, teda celú serverovú časť aj s nastavením databázy a príslušného connection stringu servera na databázu aby mohli spolu komunikovať.

Po nastavení servera a správneho poskytovania webových služieb môžete spustiť samotného klienta aplikácie UnIS. Nachádza sa na priloženom CD v oddiely (Systém UnIS\UnIS client\UnIS klient pre desktopové aplikácie\Mono Framework - Linux\RUN UnIS), kde je uložený pod názvom UnIS-MONO.exe. Rovnako ako v predchádzajúcom príklade nasadenia, je potrebné pre beh aplikácie z vášho hard disku, potrebné nakopírovať celý adresár RUN UnIS, ktorý obsahuje skompilované kódy, spustiteľne pod Mono Framework. Samotné spustenie klienta DA pod Mono v prostredí Windows je nasledovné:

- Spustite príkazovú riadku Mono Command Prompt podľa príkladu v oddiely Inštalácie a otestovania Mono pod Windows.

- Nastavte cestu k súboru UnIS for Mono a spustite ho príkazom: mono UnIS for Mono.exe.

Pri prvom spustení budú prevedené kontroly a inštalácia firmy popísané v predchádzajúcej kapitole: 6.1.1 Beh aplikácie pod .NET Framework.

6.2 Nasadenie na Windows Mobile

Súčasťou systému UnIS je klient MA pre mobilné zariadenia. Testovanie bolo prevádzané pomocou simulátora vo Visual Studiu a nasadený systém bol Windows Mobile 6.0. Tiež bol systém testovaný aj na mobilnom zariadení HP iPAQ 114 s operačným systémom Windows Mobile® 6 Classic. Táto kapitola je zameraná na nasadenie mobilného klienta UnIS práve na takéto zariadenie.

.NET Compact Framework

Spoločnosť Microsoft vytvorila Framework založený na .NET architektúre, taktiež pre mobilné zariadenia. Táto platforma má názov .NET Compact Framework a je potrebná pre beh klientskej časti nášho systému pre mobilné zariadenia. Potrebná verzia je .NET Compact Framework 3.5, ktorú je možné stiahnuť na stránkach spoločnosti Microsoft, v sekcii sťahovania softwaru: http://www.microsoft.com/downloads/en/default.aspx.

Databáza

Rovnako ako centrálna databáza systému, je základným kameňom mobilného klienta jeho vlastná databáza, ktorá beží na Microsoft SQL Server Compact (SQL CE). Jedná sa o server, ktorý sa stará o prácu s databázou uloženou do vytvoreného súboru s príponou .sdf. Tento databázový súbor sa spolu s ostatnými komponentmi potrebnými pre chod aplikácie prenáša do operačného systému klienta, kde následne do neho pristupuje UnIS Pokladnica a pracuje s ním prostredníctvom SQL CE servera. Táto databáza obsahuje aj testovacie dáta pre prvé spustenie a otestovanie klienta MA. Možnosť stiahnutia inštalačného súboru s príponou .cab je na spomínaných stránkach spoločnosti Microsoft. Tento súbor s príponou .cab zaistí nainštalovanie serveru priamo v operačnom systéme mobilného zariadenia.

Spustenie klienta MA

Pre spustenie klienta na mobilnom zaradení stačí nakopírovať do systému Windows Mobile zložku aj so súbormi, ktorá je umiestnená na priloženom CD v oddiely (Systém UnIS\UnIS client\UnIS klient pre mobilné zariadenia) a jedná sa o: UnIS klient PDA. Táto časť obsahuje spustiteľný súbor UnIS (Mobile), ktorý spustí vytvoreného mobilného klienta a využije databázu prenesenú pri kopírovaní. Pri prvom spustení je už vytvorený užívateľ v databáze klienta s prihlasovacími údajmi (Login: admin, Heslo: pass). Pomocou tohto užívateľa sa môžete prihlásiť do systému. Ak však budete chcieť využívať vlastné prihlasovacie údaje, využívané v klientovi DA a uložené v hlavnej centrálnej databáze, potom musíte spustiť synchronizáciu užívateľov ešte pred prihlásením. Podmienkou je však správne nastavenia pripájacieho reťazca k webovým službám poskytovaným serverom.

6.3 Nasadenie na Linux

Nasadenie vytvoreného systému UnIS v prostredí Linux, tak aby bežal na Mono Framework, si vyžaduje podrobnejšie nastavenia ako pri nasadení na operačnom systéme Windows. Najskôr spomenieme niektoré možnosti ako rozbehnúť serverovú časť aplikácie a potom vyberieme jeden z najpoužívanejších spôsobov, ktorý viac priblížime. Druhá časť sa bude zaoberať výberom vhodnej databázy pre beh na Linuxe, nastavenie databázy a vytvorenie hierarchie systému UnIS v databázovom prostredí. Nakoniec ukážeme spustenie aplikácie nad takto vytvoreným a nastaveným serverom v prostredí Linux.

6.3.1 Možnosti nasadenia serverovej časti na Linux

Máte dve možnosti ako rozbehnúť našu serverovú časť aplikácie pod Mono Framework v prostredí Linux:

- **Apache hosting** – použite mod_mono, čo je modul, umožňujúci beh ASP.NET aplikácií na Apache,

- **XSP** – je to jednoduchý spôsob ako rozbehnúť našu aplikáciu, jedná sa o jednoduchý webový server napísaný v C#.

Pre umiestnenie aplikácií odporúčame použiť možnosť mod_mono, pretože vám ponúkne všetky možnosti konfigurácie a flexibilitu servera Apache.

Majte na pamäti, že pri použití XSP, máte len veľmi obmedzené možnosti. Tie vychádzajú z podpory http len do verzie http 1.0 a neumožňujú také rozsiahle nastavenia servera ako predchádzajúci hosting Apache.

ASP.NET hosting s Apache

Modul mod_mono Apache sa používa pre beh ASP.NET aplikácií s Apache webovým serverom. To znamená, že tento modul beží spolu s Apache procesom a prevádza všetky požiadavky aplikácie ASP.NET v Mono, na procesy pre Apache a naopak. Externý ASP.NET host sa nazýva mod-mono-server a je časťou XSP modulu.

Pre jeho použitie musíte stiahnuť a nainštalovať mod_mono a xsp komponenty. Mod_mono obsahuje aktuálny Apache modul a xsp obsahuje ASP.NET hosting. Oba sú dostupné na stiahnutie zo stránok projektu Mono.

Komunikácia medzi mod_mono a Apache je založená na používaní Unix soketoch, alebo TCP soketoch. Keď autohosting nevyhovie vaším požiadavkám, budete musieť na ich rozbehnutie použiť niekoľko mod_mono Apache príkazov. Tieto príkazy sa zadávajú do hlavného konfiguračného súboru Apache (často je umiestnený v /etc/httpd/conf/httpd.conf, alebo /etc/apache2).

ASP.NET hosting s XSP

XSP je nezávislý webový server napísaný v jazyku C#, ktorý môžete použiť na rozbehnutie našej serverovej časti s minimálnym úsilím. XSP pracuje pod oboma platformami Mono aj Microsoft, preto je možné využitie spustenia klientskej aj serverovej časti aplikácie pod Mono Framework na systéme Windows. Kód je prístupný na webových stránkach projektu Mono.

Najjednoduchší spôsob spustenia XSP je z koreňového adresára aplikácie. Bude vykonávať požiadavky na porte 8080 [9].

6.3.2 Hosting s Apache serverom

Ako sme už spomenuli, Mod_Mono je modul pre Apache 2.0/2.2. Vybrali sme si tento spôsob využitia servera Apache pre našu serverovú časť aplikácie UnIS hlavne kvôli obľúbenosti servera Apache, dostupného z webovej adresy: http://httpd.apache.org, a možnosti behu na platforme Unix.

Požiadavky k nastaveniu serverovej časti UnIS týmto spôsobom sú nasledovné:

- Budete potrebovať nainštalovať webový server Apache dostupný z vyššie spomenutých stránok.

- Druhou podmienkou použitia je získanie mono, xsp a mod_mono z oficiálnych stránok projektu Mono: http://www.go-mono.com/mono-downloads/download.html.

V závislosti na distribúcii verzie systému Linux sa môžu niektoré nastavenia líšiť. Preto je vhodné používať dokumentáciu vašej distribúcie, pred skompilovaním zdrojových kódov [12].

Jednoduchá konfigurácia Mod_Mono

Po nainštalovaní XSP sa zároveň vytvoria niektoré príklady ASP.NET stránok s nastavenými webovými službami. Ak slúži ku konfigurácií XSP prefix /usr, ukážkové súbory sa umiestnili do (/usr/lib/XSP/test).

Ak nevyžadujete nejaké komplikované nastavovanie servera Apache, jednoducho stačí použiť AutoHosting, čo v podstate znamená načítanie mod_mono.conf súboru do konfiguračného súboru Apache:

```
        Include /etc/apache2/mod_mono.conf
```

Aplikácia bude pripravená na použitie. Ak to chcete vyskúšať, nakopírujte adresár (/usr/lib/xsp/test) do domovského adresára Apache v openSUSE (/srv/www/hdtocs).

Odporúčame vytvoriť nový adresár, pre testovanú aplikáciu. To vám umožní pomocou xcopy preniesť aplikáciu z Windows na Linux.

Nástroje na konfiguráciu Mod_Mono

Ak opísaný spôsob nastavenia servera pomocou AutoHosting nevyriešil správne spustenie a nastavenie servera, musíte vložiť niekoľko mod_mono Apache adresárov do hlavného konfiguračného súboru Apache servera a tým zaistiť správny beh serverovej časti UnIS systému.

Apache mod_mono configuration tool je nástroj umožňujúci generovanie nastavenia pre Virtual Hosts a nastavenia pre ASP.NET aplikácie. Príklad použitia tohto nástroja nájdete na webových stránkach projektu Mono: http://go-mono.com/config- mod-mono/.

V najjednoduchšom prípade, nemali by ste musieť zadávať nič iné, len server a názov aplikácie, teda nastavenie ciest k umiestneniu aplikácie serverovej časti UnIS. Cieľom nástroje je nastavenie mod_mono a zladenie s Mono a platformou na ktorej chcete server spustiť.

Po vyplnení formulára bude vygenerovaný konfiguračný súbor, ktorý môžete uložiť na disk. Aby ste otestovali nastavenie a beh servera, potom bude potrebné Apache server reštartovať príkazom:

```
sudo /sbin/service apache2 restart
```

6.3.3 Databáza MySQL

Ako databázový server vhodný pre spustenie na Linux, sme si vybrali veľmi populárny open source databázový server MySQL.

Požiadavky k spusteniu databázy na servery MySQL:

- **MySQL Server** – jedná sa o open source databázový server, podporujúci množstvo platforiem. Získať ho môžete z adresy: http://www.mysql.com/downloads/mysql/#downloads,

- **MySQL Connector / Net** – slúži pre komunikáciu aplikácie s databázou a nevyžaduje klientsku knižnicu. Je poskytovaný pod licenciou GPL a odporúčame ho použiť s Mono. Dostupný je na adrese: http://dev.mysql.com/support/.

Formát connection string, ktorý je potrebné nastaviť v serverovej časti UnIS aplikácie je nasledovný:

```
"Server=hostname;" + "Database=database;" + "User ID=username;" +
"Password=password;" + "Pooling=false"
```

Príklad odlišnosti od vytvoreného serverového prístupu k dátam pomocou MSSQL je zobrazený na nasledujúcom zdrojovom kóde. Preto pri nasadení na odlišnú databázu, ako pre ktorú bola aplikácie vytvorená, je potrebné prepísanie serverovej časti za pomoci využitia (using MySQL.DATA.MySqlClient) [15].

```
using System;
using System.Data;
using MySql.Data.MySqlClient;

public class Test
{
        public static void Main(string[] args)
        {
                string connectionString =
                "Server=localhost;" +
                "Database=test;" +
                "User ID=myuserid;" +
                "Password=mypassword;" +
                "Pooling=false";
                IDbConnection dbcon;
                dbcon = new MySqlConnection(connectionString);
```

```
                        dbcon.Open();
                        IDbCommand dbcmd = dbcon.CreateCommand();

                        string sql = "SQL parametrizovaný dotaz";
                        dbcmd.CommandText = sql;
                        IDataReader reader = dbcmd.ExecuteReader();
                        while(reader.Read()) {
                                string test = (string) reader["StÍpec"];
                        }

                        reader.Close();
                        reader = null;
                        dbcmd.Dispose();
                        dbcmd = null;
                        dbcon.Close();
                        dbcon = null;
                }
        }
```

Zdrojový kód 17: Vytvorenie spojenia a získanie dát momocou MySQL

Pre načítanie hierarchie aj s testovacími dátami je možné použiť skript, ktorý tieto tabuľky pre databázu UnIS systému vytvorí a je uložený na priloženom CD v oddiely (Databáza\MySQL\Skript pre vytvorenie DB) s názvom MySQL_UnIS.sql. Pre uľahčenie nastavenia databázy odporúčame použiť MySQL Administrátor, kde pomocou nástroja MySQL Query Browser môžete načítať vytvorený skript a prevedení vytvoriť hierarchiu centrálnej databázy UnIS systému.

Spustenie testovacej verzie systému UnIS s naplnenou databázou, môžete uskutočniť prostredníctvom druhého poskytované skriptu na priloženom CD v oddiely (Databáza\MySQL\Skript pre naplnenie DB testovacími dátami) pod názvom DB_Test_Data_MySQL_UnIS.sql. Tento skript nielen vytvorí hierarchiu databázy, ale aj naplní tabuľky testovacími dátami pre uľahčenie spustenia klienta a vyhnutie sa jednotlivých krokov inštalácie UnIS aplikácie pri prvotnom spustení.

6.3.4 Spustenie klienta DA

Ešte pred spustením klientskej časti systému je potrebné mať správne vytvorenú databázu, ktorá má obsahovať celkovú hierarchiu systému UnIS. Výber vhodného databázového servera, postup pri jeho spustení a vytvorenie databázy je popísaný v predchádzajúcej kapitole. Serverovú časť však tvorí hlavne server, ktorý poskytuje webové služby a prostredníctvom nich komunikuje klient s databázou systému. Za vhodný server na beh v prostredí Linux sme zvolili Apache a nestavenia sme opísali v kapitole: 6.3.26.3.2 Hosting s Apache serverom. Potrebné dáta k nastaveniu servera Apache s Mod_Mono nájdete na priloženom CD v oddiely (Systém UnIS\UnIS server\Mono Framework - Linux).

Po správnom nastavení databázového a webového servera je rad na spustení samotného klienta pod Mono Framework v prostredí Linux. Aby ste mohli bezproblémovo spustiť klienta DA pre desktopové aplikácie, poskytli sme všetky potrebné skompilované súbory na priložené CD v oddiely (Systém UnIS\UnIS client\UnIS klient pre desktopové aplikácie\Mono Framework -

Linux\RUN UnIS), kde sa nachádza Adresár s názvom RUN UnIS, ktorý je potrebné nakopírovať do prostredia Linux v ktorom bude klient spúšťaný. Tak ako v každej verzii nasadenia, opäť pripomíname, že je potrebné spúšťať klienta DA z tohto adresára, ktorý obsahuje všetky potrebné komponenty pre jeho správny beh. Spôsoby spustenia klienta sú podobné ako v prostredí Windows, teda spustenie exe súboru UnIS-MONO.exe pomocou príkazovej riadky Mono Command Prompt. Príkaz je už známy: mono UnIS-MONO.exe.

Prvé spustenie bude záležať od stavu vytvorenej databázy, teda od voľby vytvorenia hierarchie naplnenej testovacími dátami, alebo nenaplnenej testovacími dátami. Pri testovacích dátach nebude vyvolávaná inštalácia systému, ale bude systém poskytovať prihlásenie vytvoreného testovacie užívateľa do pozície administrátora. Prihlasovacie meno v takomto prípade je nastavené na hodnotu „admin“ a heslo na hodnotu „pass“. V prípade ostrého spustenia aplikácie bez testovacích dát, si administrátora s plným prístupom vytvárate vyplnením príslušného formulára a to isté platí pre firmu, v ktorej bude systém nasadený. Inštalácie je názorne zobrazená v prílohe: B Inštalácie systému UnIS.

7 Záver

S postupom času si ľudia začínajú uvedomovať, že pre život v súčasnej spoločnosti je potrebné vedieť sa rýchlo orientovať vo svete informácií. Práve informácie a ich spracovávanie prostredníctvom aplikačného softvéru boli predmetom našej diplomovej práce.

Cieľom tejto práce bolo navrhnúť a vytvoriť univerzálny informačný systém, ktorý uľahčí a zefektívni prácu v oblasti obchodu. Zvýšenú pozornosť sme venovali najmä prenosu aplikácie medzi rôznymi platformami, čo nám umožnila platforma Mono Framework.

Práca bola ponímaná tak, aby slúžila predovšetkým v reštauračných zariadeniach, ktoré nedisponovali žiadnym informačným systémom a nevyhnutne ho potrebovali. Keďže sa jedná o aplikáciu, ktorá pokrýva rozsiahlu oblasť reštauračného predaja s vlastným skladovým hospodárstvom je možné aplikáciu ďalej rozšíriť o ďalších mobilných klientov. Títo budú môcť vytvárať objednávky a teda taktiež na diaľku spravovať celé skladové hospodárstvo. Ďalšou alternatívou je rozšírenie klientskej časti systému o čítačku čiarových kódov.

Systém bol otestovaný v reštaurácii zadávateľa a z reakcií zadávateľa a užívateľov môžeme skonštatovať, že požiadavky na vytvorenie daného systému boli splnené. Za najväčší úspech našej aplikácie považujeme priaznivú reakciu zákazníkov, ktorí prišli so systémom do kontaktu v podobe vytvorených účtov, faktúr atď.

Námet na vytvorenie tohto projektu ma zaujal hneď v počiatku, keď som z vlastnej skúsenosti spoznal náročnosť ručného spracovania množstva informácií bez použitia informačného systému. Preto po návrhu vytvoriť systém pre obchod a sklad som neváhal a rozhodol sa uľahčiť prácu ľuďom, pre ktorých je oveľa náročnejšie udržať si poriadok vo vlastnej evidencii bez elektronického spracovania dát. Preto dúfame, že navrhnutý systém bude plne využívaný a jeho funkcie dopomôžu ku skvalitneniu služieb, ponúkaných zákazníkom v nami vybranej oblasti.

8 Literatúra

[1] DUMBILL, Edd - BORNSTEIN, Niel M. *Mono: A Developer's Notebook*. O'Reilly Media, 2004. ISBN 0-596-00792-2.

[2] Erza, Aviad. *Model View Presenter (MVP) Design Pattern with .NET - Winforms vs. ASP.NET Webforms* [online]. c200810, posledná verzia 29.4.2010 [cit. 2010-5-1]. Dostupné z: <http://aviadezra.blogspot.com/2008/10/model-view-presenter-design-pattern.html>.

[3] Erza, Aviad. *MVC (Model View Controller) Design Pattern* [online]. c200806, posledná verzia 29.4.2010 [cit. 2010-5-1]. Dostupné z: <http://aviadezra.blogspot.com/2008/06/mvc-model-view-controller-design.html>.

[4] Ecma-international. *C# Language Specification* [online]. c200606, posledná verzia 22.7.2008 [cit. 2010-4-15]. Dostupné z: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>.

[5] Ecma-international. *Standard ECMA-334* [online]. c200606, posledná verzia 13.7.2009 [cit. 2010-4-10]. Dostupné z: < http://www.ecma-international.org/publications/standards/Ecma-334.htm>.

[6] Ecma-international. *Standard ECMA-335* [online] c200606, posledná verzia 13.7.2009 [cit. 2010-4-10]. Dostupné z : < http://www.ecma-international.org/publications/standards/Ecma-335.htm>.

[7] Mike. *Getting Started with UML* [online]. c1997, posledná verzia 15.12.2010 [cit. 2010-4-10]. Dostupné z: < http://www.uml.org/>.

[8] Mono-project. *Application Portability* [online], posledná verzia 4.6.2007 [cit. 2010-4-10]. Dostupné z: <http://mono-project.com/Guidelines:Application_Portability>.

[9] Mono-project. *ASP.NET* [online], posledná verzia 25.3.2010 [cit. 2010-5-1]. Dostupné z: <http://www.mono-project.com/ASP.NET>.

[10] Mono-project. *General* [online], posledná verzia 4.12.2009 [cit. 2010-4-15]. Dostupné z: <http://www.mono-project.com/FAQ:_General>.

[11] Mono-project. *Licensing* [online], posledná verzia 29.4.2010 [cit. 2010-4-15]. Dostupné z: <http://mono-project.com/License>.

[12] Mono-project. *Mod mono* [online], posledná verzia 17.1.2010 [cit. 2010-4-10]. Dostupné z: <http://www.mono-project.com/Mod_mono>.

[13] Mono-project. *Mono early history* [online]. c2003[10], posledná verzia 15.10.2003 [cit. 2010-4-15]. Dostupné z: <http://lists.ximian.com/archives/public/mono-list/2003-October/016345.html>.

[14] Mono-project - Icaza, Miguel. *MoMA - Issue Descriptions* [online], posledná verzia 12.3.2007 [cit. 2010-5-1]. Dostupné z: <http://www.mono-project.com/MoMA_-_Issue_Descriptions>.

[15] Mono-project. *MySQL* [online], posledná verzia 25.3.2010 [cit. 2010-5-1]. Dostupné z: <http://www.mono-project.com/MySQL>.

[16] Mono-project - Pobst, Jonathan. *Porting Winforms Applications* [online], posledná verzia 16.6.2007 [cit. 2010-5-1]. Dostupné z: <http://mono-project.com/Guide:_Porting_Winforms_Applications>.

[17] Mono-project. *Roadmap History* [online], posledná verzia 21.1.2009 [cit. 2010-4-15]. Dostupné z: <http://www.mono-project.com/Roadmap_History>.

[18] Mono-project. *Supported platforms* [online], posledná verzia 12.2.2009 [cit. 2010-4-10]. Dostupné z: <http://mono-project.com/Supported_Platforms>.

[19] Mono-project. *Using Mono on Windows* [online], posledná verzia 30.3.2010 [cit. 2010-4-10]. Dostupné z: <http://www.mono-project.com/Mono:Windows>.

[20] Mono-project. *What is Mono* [online], posledná verzia 12.2.2009 [cit. 2010-4-10]. Dostupné z: < http://mono-project.com/What_is_Mono/>.

[21] Paul Yao, Paul – Durant, David. *SQL Server CE: New Version Lets You Store and Update Data on Handheld Devices* [online]. United States: MSDN magazine, 2001. [cit. 2010-5-1]. Dostupné z: <http://msdn.microsoft.com/en-us/magazine/cc301933.aspx>.

[22] PRICE, Jason. *C# programování databází*. Preložil Vilém Vrbický. 1.vyd. Praha: Grada, 2005. Úvod, s. 27-28. ISBN 80-247-0982-1.

[23] Reese, George. *Distributed application architecture* [online]. c2000[11], posledná verzia 18.11.2007 [cit. 2010-4-15]. Dostupné z: <http://java.sun.com/developer/Books/jdbc/ch07.pdf>.

[24] RICHTER, Jeffrey. *.NET Framework programování aplikací*. Preložil Jiří Hynek. 1.vyd. Praha: Grada, 2003. ISBN 80-247-0450-1.

[25] Wells, Jonathan. *An Introduction to P/Invoke and Marshaling on the Microsoft .NET Compact Framework* [online]. c2003[03], posledná verzia 29.4.2010 [cit. 2010-5-1]. Dostupné z: <http://msdn.microsoft.com/en-us/library/Aa446536>.

9 Zoznam príloh

A. Obsah CD

Priložený CD nosič obsahuje:

- Diplomová práca v elektronickej podobe (formát PDF).
- Zdrojové kódy vytvoreného informačného systému UnIS, dokumentáciu a spúšťacie súbory pre rôzne spôsoby nasadenia klientov a servera.
- Databázové súbory pre vytvorenie hierarchie databázy aj s testovacími dátami.

Systém UnIS

Zložka Systém UnIS obsahuje programátorské príručky dvoch typov. Jedným je vytvorenie príručiek formátu MSDN a druhým je formát webovej prezentácie, kde je zhrnutý celý vytváraný systém. Na vytvorenie sme využili software NDoc, pomocou ktorého sme vygenerovali príručky z .NET komentárov použitých vo vytvorenom kóde. Dostupnosť tohto softwaru je na adrese: http://ndoc.sourceforge.net/ v sekcii Downloads.

Táto zložka obsahuje aj kompletné projekty celého systému UnIS do vývojových nástrojov Microsoft Visual Studio 2008 a tiež systém UnIS okresaného o knižnice na vytváranie PDF výstupov (z dôvodu nekompatibility s Mono Framework), ako projekty do vývojových nástrojov Microsoft Visual Studio 2008 a Mono Develop 2.2.2. Jednotlivé zdrojové kódy sú skompilované a umožňujú spustenie aplikácie priamo z ich adresára (bin).

Keďže systém je rozdelený na klientsku a serverovú časť, aj zložky na priloženom CD v tomto adresári sú rozdelené na spustenie a nastavenie servera, či klientov. Zložka (UnIS client) je ďalej rozdelená na nasadenie klientov pre desktopové aplikácie, alebo druhého typu klienta MA pre mobilné aplikácie. Serverovú časť reprezentuje zložka (UnIS server), ktorá obsahuje zložky podľa nasadenia servera na platformu Linux, alebo Windows a tiež podľa druhu skompilovaného kódu v .NET Framework, alebo Mono Framework. Jednotlivé rozmiestnenie klientov a spôsob nastavenia serverov s databázou je popísaný v kapitole: 6 Nasadenie systému UnIS.

Databáza

Zložka databázy je rozdelená na dve časti. Jedná obsahuje skripty pre Microsoft SQL Server a druhá pre MySQL Server. Skripty pre každý server sú dvoch typov, teda skripty pre vytvorenie databázy a skripty pre vytvorenie databázy aj s testovacími dátami pre urýchlenie spustenia aplikácie (bez prvotnej inštalácie systému UnIS, s vytvorenou testovacou firmou a testovacími užívateľmi). Rovnako ako nastavenie klientov, aj nastavenie databázy a servera je popísané v kapitole: 6 Nasadenie systému UnIS.

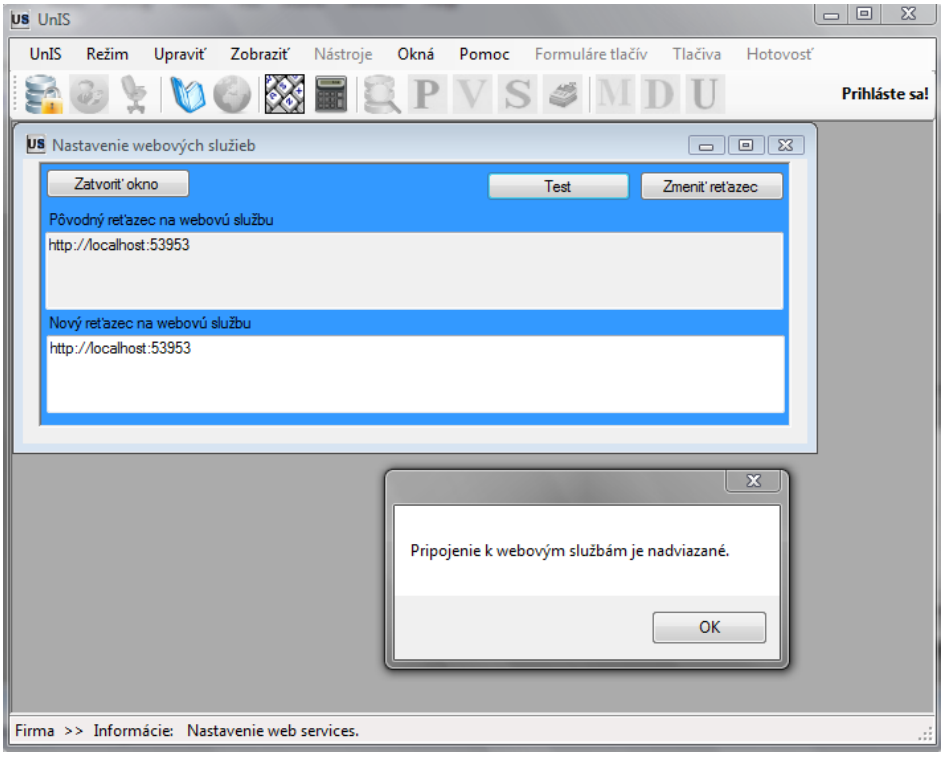
B. Inštalácie systému UnIS

Obsahom tejto prílohy je stručný popis inštalácie systému na zobrazených obrázkoch. Inštaláciou sa začína prvotné spustenie klienta DA systému UnIS. Teda ak nie je ešte v databáze vytvorená firma a administrátor, ktorý ju bude spravovať. Len pre zaujímavosť chceme poznamenať, že systém neumožňuje vymazanie vytvoreného administrátora, ktorý je práve prihlásený.

Však ešte pred samotnou inštaláciou systému je potrebné aplikáciu UnIS správne nasadiť na požadovanú platformu a dostupný operačný systém. Po nastavení databázového servera je nevyhnutné prispôsobenie connection string reťazca na vytvorenú databázu v serverovej časti aplikácie. Ak spojenie servera s databázou nebude nadviazané, nepodarí sa vám správne spustiť aplikáciu a teda ani nainštalovať a nastaviť prvotné kroky pre použitie opísané nižšie.

Nastavením databázy a servera ešte nekončí úplné nastavenie prostredia informačného systému na prácu s databázovými záznamami. Je potrebné nastaviť reťazec na komunikáciu so serverom, teda s poskytovanými webovými službami cez ktoré bude server s klientmi komunikovať. To znamená nastavenie a otestovanie webovej adresy a server.

Nasledujúci obrázok predstavuje nastavovanie adresy na pripojenie klienta k webovým službám servera, ktorý je umiestnený na adrese: http://localhost, a komunikuje cez port 53953.



Obrázok 37: Inštalácia UnIS - Nastavenie reťazca k poskytovaniu webových služieb

Ale teraz k samotnej inštalácií, vychádzajúcej z nasledujúcich obrázkov:

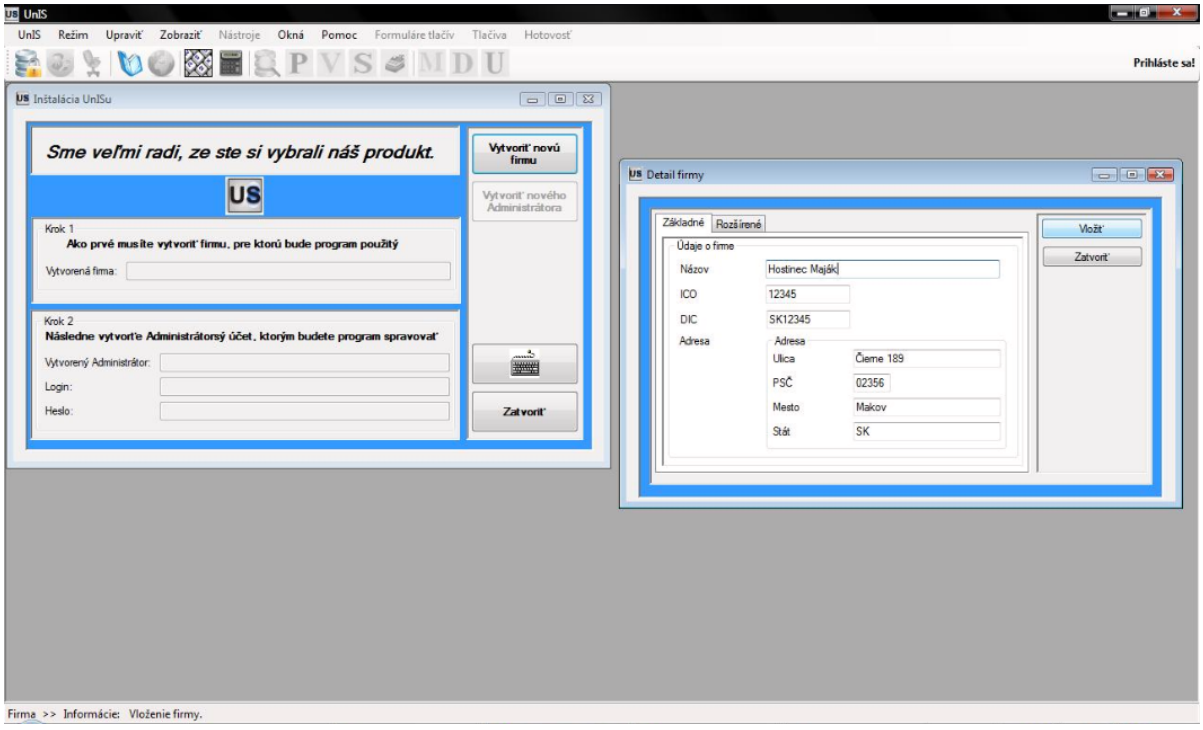
- **Vytvorenie firmy** - pri stlačení tlačidla (Vytvoriť novú firmu) sa zobrazí formulár pre vytvorenie záznamu o detailných informáciách firmy. Po vložení firmy sa v inštalačnom okne objaví názov zadanej firmy a povolí sa stlačenie tlačidla (Vytvoriť nového Administrátora),

- **Vloženie administrátora** - pri stlačení tlačidla (Vytvoriť nového Administrátora) sa zobrazí formulár pre zaregistrovanie administrátora firmy. Po vložení administrátora sa zobrazia údaje v inštalačnom okne a užívateľ môže okno zatvoriť. Inštaláciou bola vytvorená firma a administrátor,

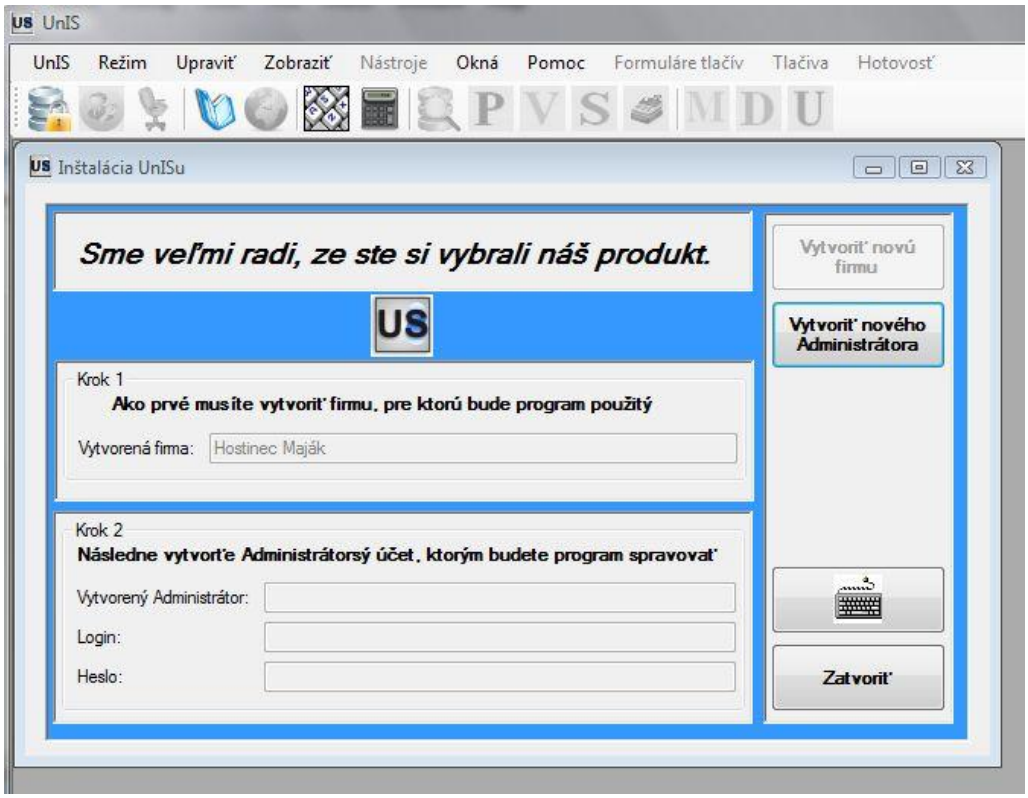
- **Otvorenie firmy** – teraz máte možnosť otvoriť vytvorenú firmu v inštalácií,

- **Prihlásenie užívateľa** – prihláste sa loginom a heslom, použitým pri inštalácií UnIS,

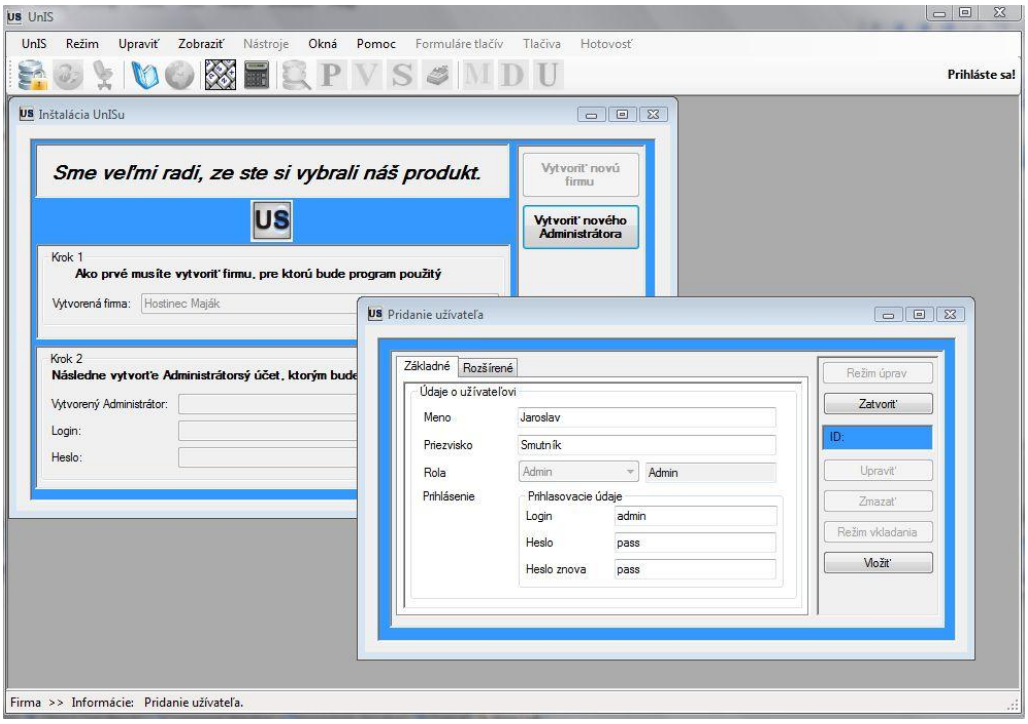
- **Administrátorský režim** – jeho spustenie je automatické, po prihlásení do systému.



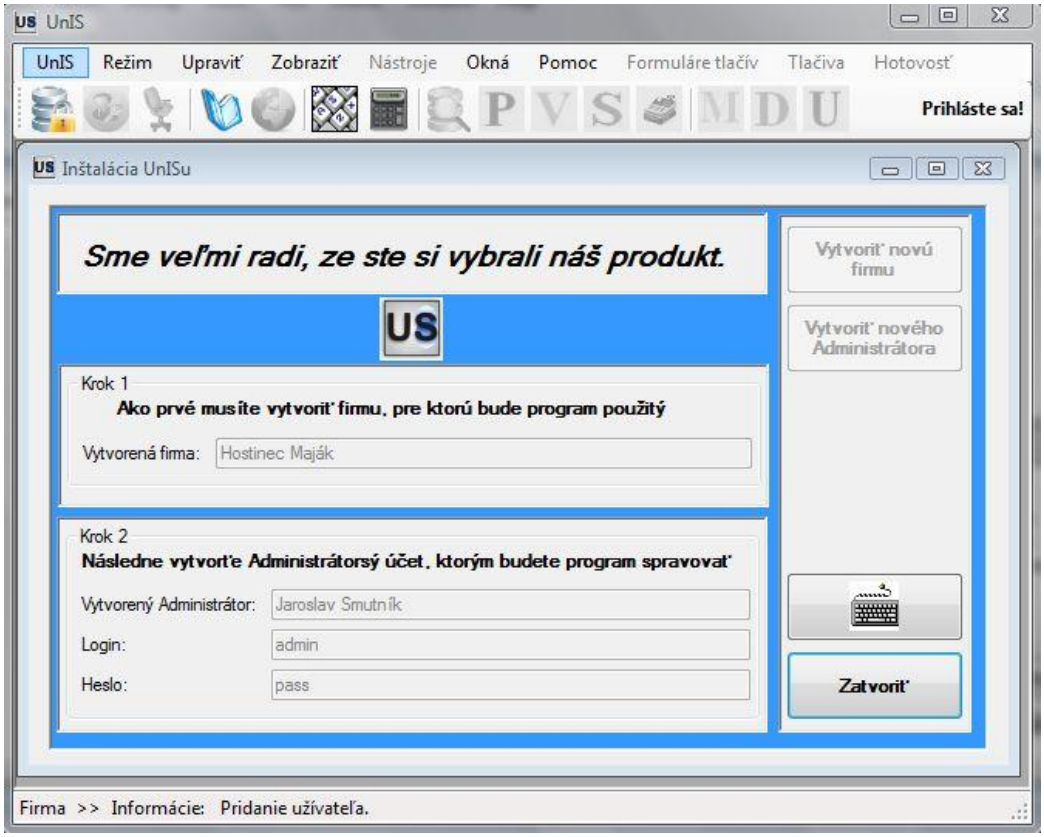
Obrázok 38: Inštalácia UnIS - Vytvorenie firmy



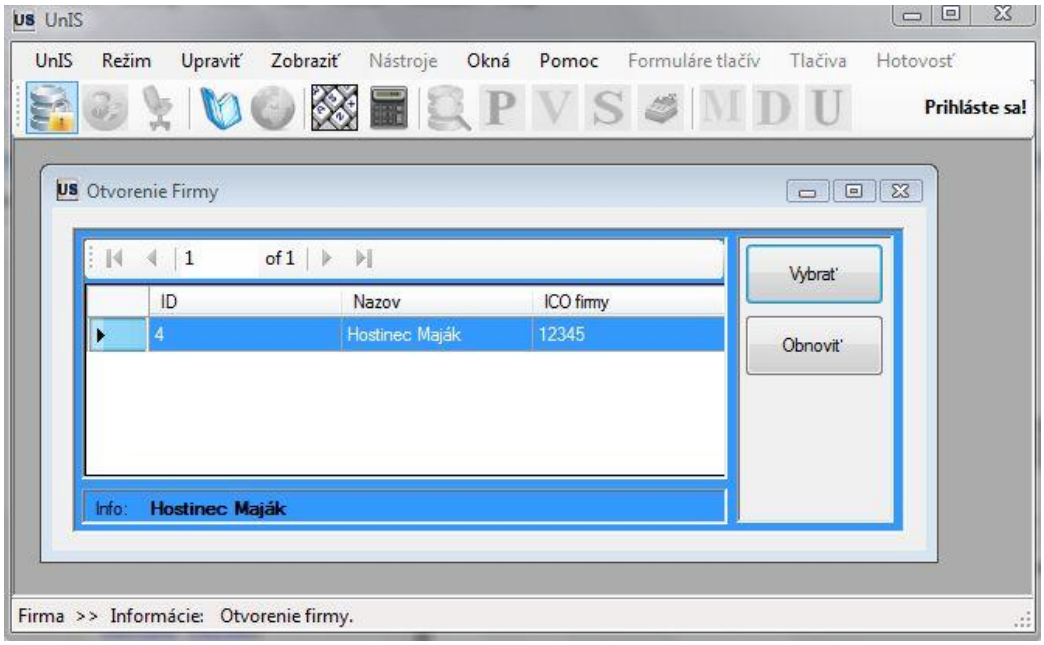
Obrázok 39: Inštalácia UnIS - Vytvorená firma



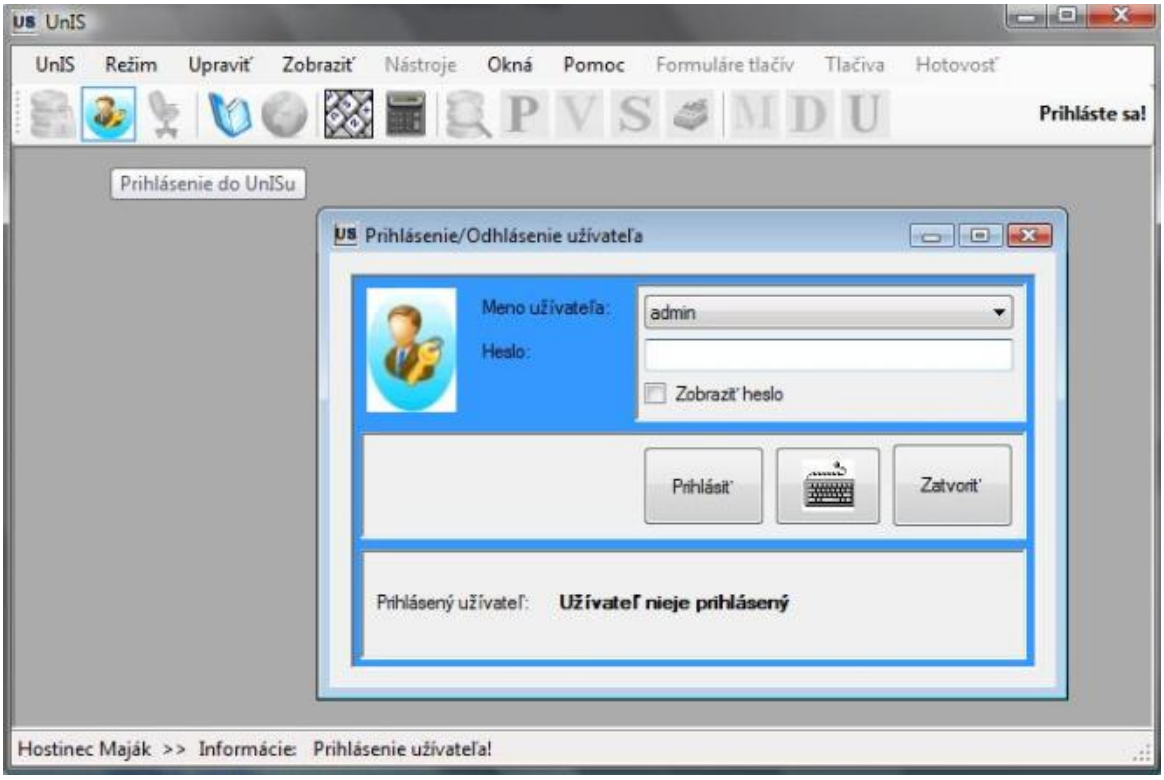
Obrázok 40: Inštalácia UnIS - Vloženie administrátora



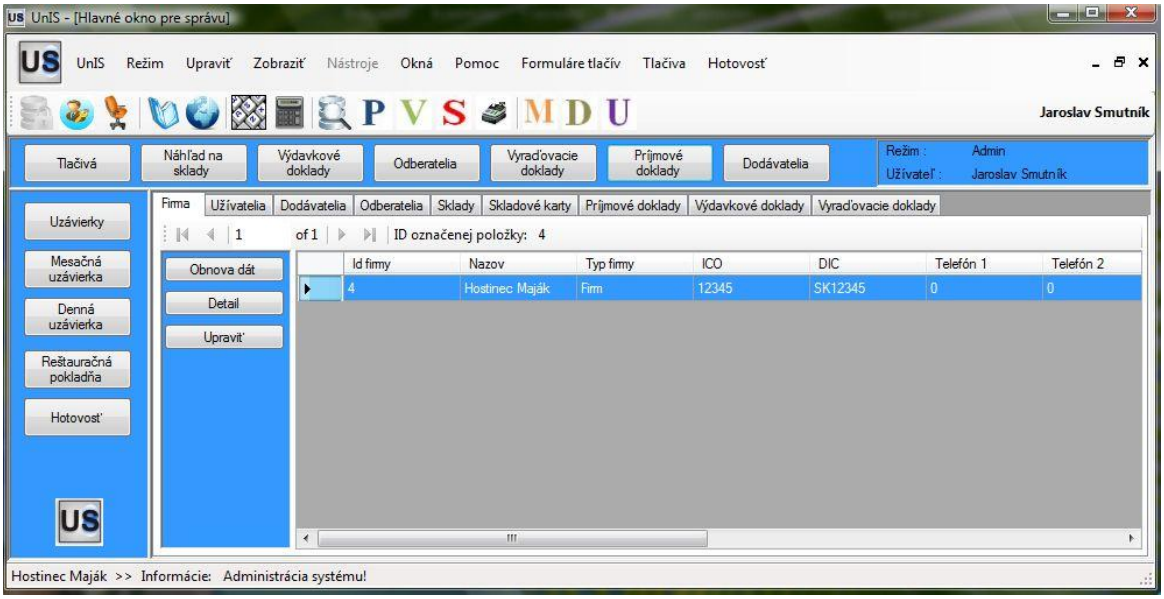
Obrázok 41: Inštalácia UnIS - Inštalácia dokončená



Obrázok 42: Inštalácia UnIS - Otvorenie firmy



Obrázok 43: Inštalácia UnIS - Prihlásenie užívateľa



Obrázok 44: Inštalácia UnIS - Administrátorský režim

C. Výstupy a obrázky

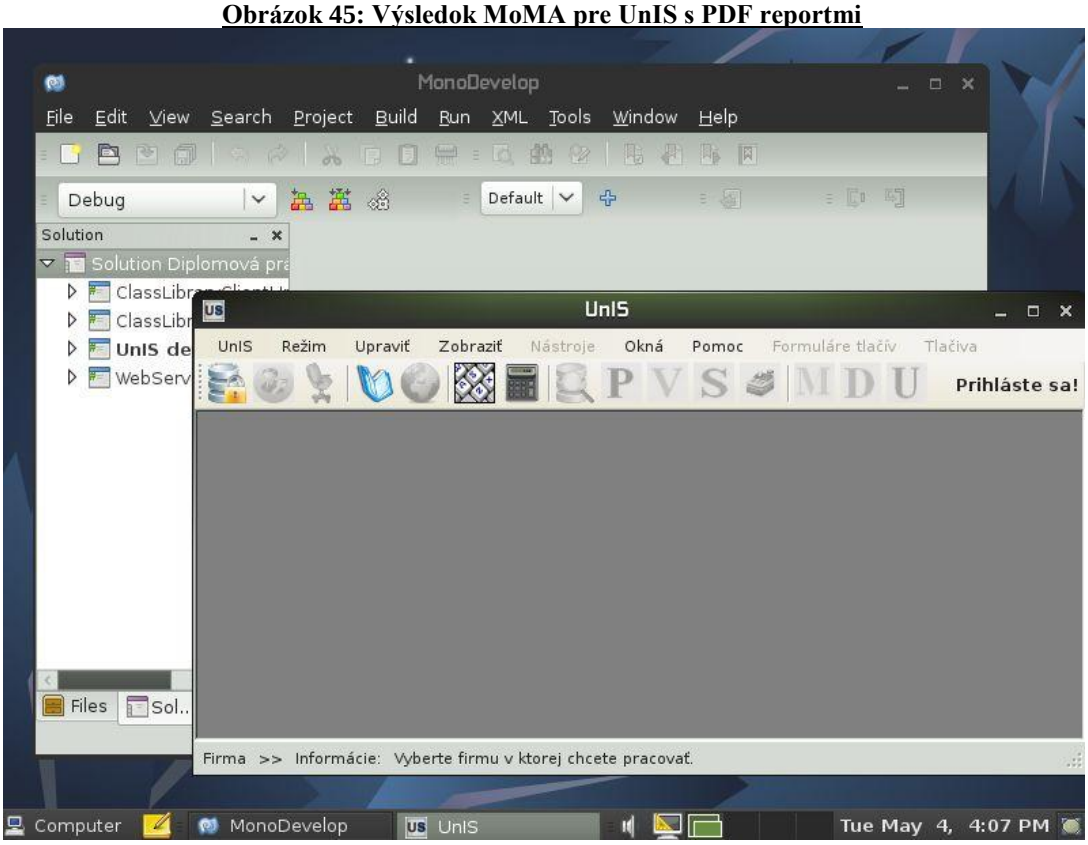
MoMA Scan Results

Scan Date: 27. 4. 2010 1:05:20
MoMA Definitions: Mono 2.6

For descriptions of issues, see MoMA Issue Descriptions.

Assembly	Version	Missing	Not Implemented	Todo	P/Invoke
ClassLibraryClientUnIS.dll	1.0.0.0	0	0	0	0
ClassLibraryServerUnIS.dll	1.0.0.0	0	0	0	0
Interop.Acrobat.dll	1.1.0.0	0	0	0	0
Interop.AcrobatAccessLib.dll	3.0.0.0	0	0	0	0
itextsharp.dll	5.0.0.0	0	0	0	0
MigraDoc.DocumentObjectModel.dll	1.31.3066.0	0	0	0	0
MigraDoc.Rendering.dll	1.31.3066.0	0	0	0	2
MigraDoc.RtfRendering.dll	1.31.3066.0	0	0	0	0
PdfSharp.Charting.dll	1.31.1789.0	0	0	0	0
PdfSharp.dll	1.31.1789.0	0	0	8	22
UnIS (Windows).exe	1.0.0.0	0	0	1	0
UnIS (Windows).vshost.exe	9.0.0.0	0	0	0	0
WebServiceUnIS.dll	1.0.0.0	0	0	0	0
Totals		0	0	9	24

Obrázok 45: Výsledok MoMA pre UnIS s PDF reportmi



Obrázok 46: Spustenie UnIS v prostredí Linux